Building a Shared Ontology Use Patterns Repository

Jonathan P. Bona*[1], Joseph Utecht[1], Sarah Bost[1], Corey J. Hayes[1,2,3], Mathias Brochhausen[1]

[1]*Department of Biomedical Informatics, University of Arkansas for Medical Sciences, Little Rock, AR, USA*
[2]*Department of Psychiatry, University of Arkansas for Medical Sciences, Little Rock, AR, USA*
[3]*Center for Mental Healthcare and Outcomes Research, Central Arkansas Veterans Healthcare System, Little Rock, AR, USA*
jpbona@uams.edu, jutecht@uams.edu, sjbost@uams.edu, cjhayes@uams.edu, mbrochhausen@uams.edu

Abstract

This paper proposes and reports on the creation of a biomedical ontology use patterns repository. *This work aims to facilitate the curation, sharing, discovery, and use, of information about how OBO and other biomedical ontologies are used by informatics researchers and other ontology users to transform biomedical instance data into realist semantic representations. By encouraging sharing of information about how ontologies are actually used with instance data this resource will reduce the total effort by ontology users to design and implement representations for use with their data. We believe this will ultimately result in more widespread use of higher-quality representations, and in improved semantic interoperability of data. Our repository proof of concept implementation is available as a web application built and organized using semantic web technologies.*

Keywords:

Ontology reuse, semantic interoperability, biomedical data

Introduction

In our biomedical knowledge representation work we often need to transform instance data that originates as entries in tabular files or other representations into semantically enhanced knowledge graphs by instantiating ontology classes as RDF individuals, along with assertions about the relations between these individuals, in a triple store database. This transformation greatly enhances the usefulness of the underlying information by making its meaning explicit, and by making it available for querying and reasoning. The benefits of this approach are well-known in the biomedical ontologies community: by using axiomatically-rich realist ontologies that share a common upper level based on a shared theory of reality, we can generate consistent representations for data that are trivially interoperable, and include machine-accessible semantics that allow semantic web reasoners and related tools to infer new information based on the represented instances.

This approach is useful both as the basis for semantic representations used for newly collected/generated data that can be instantiated automatically by ontology-based software systems, including our efforts in *Comparative Assessment Framework for Environments (CAFE) of Trauma Care (1)*, and in the *Data Coordinating and Operations Center (DCOC) for the IDEAS States Pediatric Clinical Trials Network*. It is useful as well for enhancing and integrating pre-existing data, or other data whose ongoing generation beyond the control of ontologists, for instance in our work on the *Platform for Imaging in Precision Medicine (PRISM)* initiative (2) and related ongoing projects.

In collaborations of multidisciplinary teams, especially with collaborators who are not accustomed to using ontology-driven knowledge representation strategies, we sometimes encounter the initial expectation that in order to build semantic representations for instance data, it is necessary only to select the single ontology term that best matches the meaning for each column in a spreadsheet of clinical data, for instance. In fact, such one-to-one mappings are rarely possible or desirable given the complexity of the world (which these data are supposed to be about) and the corresponding complexity of realist ontologies that have been carefully designed to represent the relevant portions of this reality.

For example, a *positive hpv diagnosis* might appear as a plus sign, or as the value 'true' or similar, in a column labeled 'hpv status' or similar in a table of clinical and other non-image data uploaded with a collection of *head and neck cancer* images in a cancer image archive (2). Our approach to representing the information that *this particular patient has been diagnosed with HPV* involves generating and asserting instances of classes from several different OBO Foundry Ontologies, and the relations among those instances, applying an instantiation pattern for each record, for instance the pattern shown in **Fig 1**. This pattern represents the human being (who is infected with HPV), the instance of HPV disease that inheres in that person, as well as the diagnosis itself and the planned processes that produced it. Note that this example does not deal with mistaken or retracted diagnoses, though the representation used in this could be used as part of a referent tracking (3) approach to handling such complications.

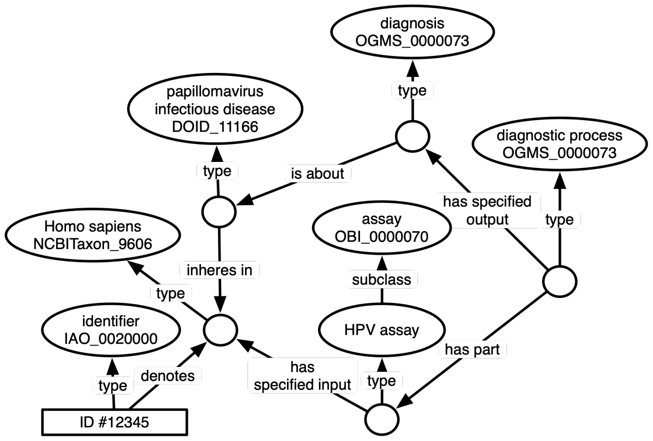


Fig. 1. Representing an instance of HPV and its diagnosis

There are many possible workflows for designing such representation patterns and applying them to instance data. In our work, we usually begin by sketching such patterns visually using a drawing program (or even a whiteboard or piece of paper), with software tools such as Protégé and Ontobee on hand to support discovering and exploring ontology terms. Once a representation pattern has been sketched out, it is manually translated into a format usable by an executable computer program that can interpret the target instance data and instantiate RDF that matches the representation pattern. This program is then run with the data as input, generating the semantically-enhanced representations suitable for use in a triple store.

Except in the small percentage of cases where this graphical depiction of the pattern is then used as an example figure in a publication, it is usually not shared outside the project, or even necessarily put in a shared space accessible to all project collaborators. Similarly, the program that realizes the RDF instantiation process is also not usually published or shared in a discoverable way, and in the best case scenario ends up in a code repository with some documentation that the project participants know how to access, but that is not easily discovered or used in other efforts where it is relevant.

One obvious issue with this practice is the duplication of effort that results when two users of OBO ontologies unwittingly work to independently represent the same, or very similar, phenomena. Even within a single group working on multiple projects, we have found it useful to have a space to share these ontology instantiation patterns. These reusable patterns consist of instances linked using `rdf:type` to the ontology classes that they instantiate, and with relations among them necessary to represent the phenomena that the patterns are about.

To the extent that there is one clear and correct way of representing instance data about a phenomenon, the biomedical ontologies community will benefit from an open repository of representational patterns for instance data that supports their publication, discussion, and reuse.

In other cases there may even be multiple possible patterns that can seem equally correct, especially where domain ontologies inadvertently overlap, or where the ontology terms used do not have definitions that completely constrain how they should be used. This will often be the case, as domain ontology developers cannot be expected to predict exactly what their ontologies will be used to represent.

One example we have encountered in our work concerns how to represent just a few of the entities involved in prescribing a drug to a patient. We have identified several possible representations that all seem to be permissible and reasonable uses of the terms involved to represent this phenomenon. These possible patterns are shown here in **Fig 2**. The first (a) has the patient (an instance of 'Homo sapiens' bearing the 'patient role') as a direct participant in the drug prescribing process. The second (b) has as its participant some instance of 'patient role,' with a path to the

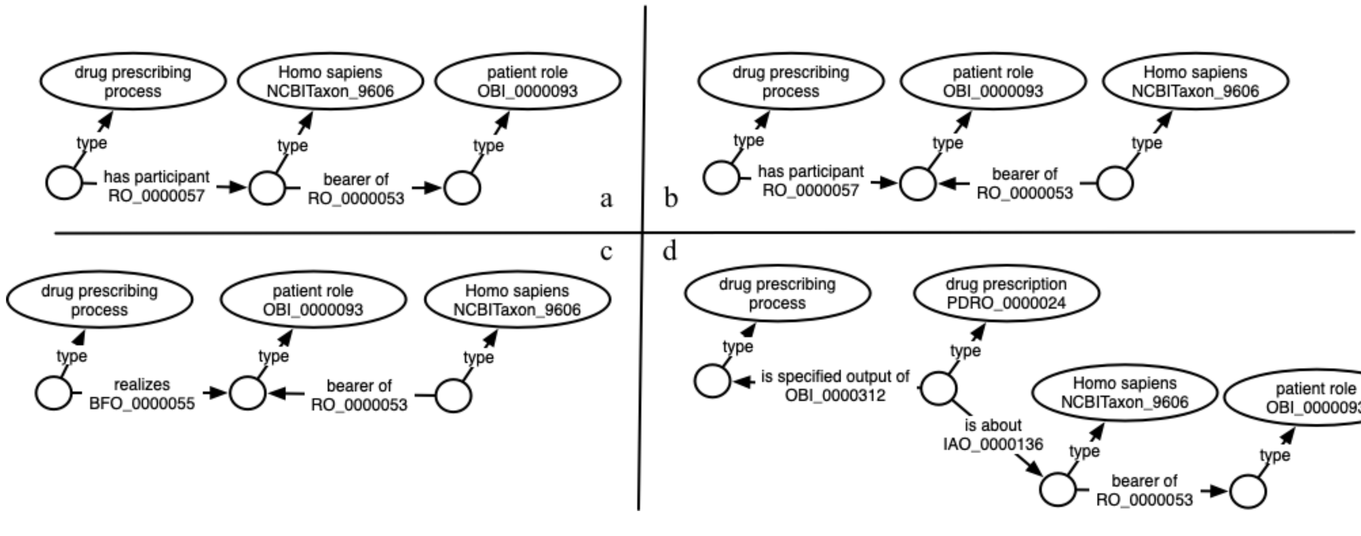


Fig. 2. Several possible patterns to represent an instance of prescribing a drug to a patient.

actual patient through their ‘bearer of’ relation to that role. While some ontology developers may assume that the participation relation only holds between occurrents and their independent continuants, this constraint is not specified in the definition of the ‘has participant’ object property. The third (c) links the drug prescribing process to the individual ‘patient role’ via the ‘realizes’ relation, which is arguably a more suitable and more informative relation to connect occurrents and dependent continuants that are realizable entities, and again links to the actual person involved through their bearing that particular role. A fourth option (d) connects the process to the role and person (patient) only indirectly through the output of the process (a ‘drug prescription’) being about the person. While this is a correct use of ‘is about’ (4), note that ‘is about’ is a fairly general relation to use here, as a prescription is certainly about several different things, including the patient who has received the prescription. Note also that (d) can be combined with the various approaches reflected in (a)-(c), resulting in even more potential representation patterns that an ontology user might decide to implement.

Related Work

The general approach of this work is related to, but still quite distinct from, the basic idea of ontology design patterns (ODPs) that has been put forward by Gangemi and Presutti (3). While the aim of this paper is to provide a pattern-based solution to the potential of different instantiation-based representations using the same ontologies in managing RDF data, the goal of ODPs is to support ontology design by providing design patterns to guide the representation of a domain in a mainly class- and object-driven manner. Recently, ODPs have become a new focus of interest in research regarding automatization of ODP creation to facilitate sharing and integration of existing ontologies (5,6).

A good example of this is Gangemi and Presutti’s agent-role pattern that aims to create a standard way to represent an agent and link it to a role (5), e.g. “John Doe” and “student role.” An example of this pattern can be found here: http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl. This patterns consists of 4 classes (in addition to owl:Thing) {Concept, Object, Role, Agent} and 4 object properties {classifies, is classified by, is role of, has role}. Notably, the pattern does not provide an instance-oriented view. While this makes perfect sense for supporting ontology development, patterns that provide insight for using ontologies to manage RDF data will necessarily include instantiations. The development of ODPs have led to the implementation of a semantic web portal for sharing and discussing ontology design patterns (7).

Our own CAFE project (1), which includes the aim to represent the organizational structures of trauma centers and trauma systems in RDF, has an internal collection of RDF instantiation patterns. These roughly 150 representations were created to model the organizations described by answers to survey questions. For example, a "yes" answer to

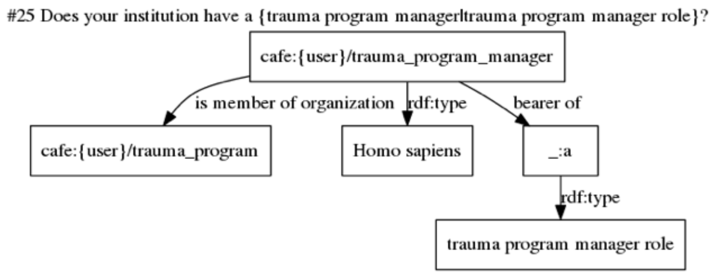

#25 Does your institution have a {trauma program manager|trauma program manager role}?
cafe:{user}/trauma_program_manager
is member of organization
rdf:type
bearer of
cafe:{user}/trauma_program
Homo sapiens
_:a
rdf:type
trauma program manager role

Fig. 3. A pattern used to generate instance data about trauma program organization in the CAFE project

the question "Does your institution have a trauma program manager?" would result in an instantiation of the triples in **Fig 3**., which shows that a human who is a member of the user’s organization is the bearer of a trauma program manager role. The CAFE project has many general instantiation patterns ranging from the simple example above to more complex representations. However, these patterns exist only in the project’s internal database with no easy way to reuse or share them.

Methods

To address this need we have implemented an ontology usage patterns repository as a web application built using a standard Python web framework combined with semantic web technology. This system provides a simple web interface that allows the user to search, browse, view, and download information about, ontology usage patterns. These pattern specifications include downloadable/reusable RDF representations, figures, textual descriptions, and other information about the pattern itself.

Implementation

Our repository application is implemented in Python 3.6 using the *Flask* web framework (8) and semantic web tools, including *rdflib* and other Python libraries. The user interface consists of HTML forms and pages rendered by Flask, with simple styling in CSS. Some of the more complex planned extensions to this work discussed more in our **Future Work** section below will require the use of Javascript for more complicated interactions.

The only database underlying this application is a triple store, which is used to persistently store all information needed to operate and organize the repository, as well as user-added information such as pattern definitions, contents, and metadata. We are currently using as a triple store a version of *Ontotext GraphDB* (9), which is a proprietary commercial triple store system available for use free of charge. However, this application will work with any system that supports SPARQL queries. The application uses the *SPARQLWrapper* Python library to query our GraphDB instance via its endpoint interface.

Application Ontology

The information required to operate this repository and represent ontology usage patterns in a triple store back-end is encoded using just RDF/OWL and a handful of classes and properties from standard semantic web resources and OBO ontologies, such as the Information Artifact Ontology (IAO). We define only a single new class, 'ontology use pattern specification' to represent the information needed to operate this repository, so technically this application is backed by a very small application ontology. That class is a subclass of IAO: 'directive information entity', as shown in **Table 1.**

In this ontology we use the persistent URL `http://purl.org/ontology-use-patterns#` as a prefix for its identifiers, including for the 'ontology use pattern specification' class, any future classes or properties that are added to achieve additional functionality, and named individuals repository triple store, for instance those instances of 'ontology use pattern specification' used to represent each individual pattern.

Name: ontology use pattern specification
URI: `http://purl.org/ontology-use-patterns#OUP_000001`
Superclass: IAO 'directive information entity' (IAO_0000033)
Definition: A directive information content entity that specifies a specific RDF representation for instance data of a particular sort using terms from pre-existing ontologies. This specification may include figures and metadata associated with the pattern in addition to RDF triples.

Table 1: ontology use pattern specification

Triple store & Named graph

The RDF triples defining each pattern added to the repository are stored within the triple store in a dedicated named graph (10) used only for that pattern. Named graphs are useful for combining information in a single triple store while maintaining some separation based on its origin, for example to assemble genomics data from different sources and manage that data along with information about its provenance in a single database (11).

By using a separate named graph for the contents of each pattern, we prevent our triple store from containing assertions that could be interpreted as making claims about the world that may be false, unverifiable, or non-referring. It should be possible, and is often desirable, to design and specify a representation pattern for future use that would contain falsehoods if it were instantiated now. An example is a pattern designed for use with some instance data that will in the future be part of a data collection that does not yet exist. In such a case, it is clearly better not to have a database that contains unconditioned assertions about those entities when they do not exist.

Leaving aside things that do not yet exist and other hypothetical entities, it is also clearly better not to have assertions to the effect that a particular instance of homo sapiens exists, that a particular instance of some disease exists, that the disease instance inheres in the human being, and so on, with instances for those individuals sitting in this repository, because this repository is not intended to store assertions about patients and their diseases. Even more practically, the system does not seek to constrain how users may choose to "name" the RDF blank node individuals that appear in their patterns (e.g. `_:person1`), though it is recommended to use names that hint at the type of individual indicated (but to never rely on this hint in place of actual type assertions). By separating out each pattern into its own named graph, we avoid the possibility of conflicts, for instance, assertions across patterns that appear to be about the same individuals. Keeping names separate is also made easier by allowing GraphDB to generate unique symbols to name individuals. Because these generated symbols are long and unwieldy for users to deal with, we

```
PREFIX oup: <http://purl.org/ontology-use-patterns#>
oup:pattern_000001 rdf:type oup:OUP_000001 .
oup:pattern_000001 rdfs:label "An example pattern" .
_:fig1 rdf:type <http://purl.obolibrary.org/obo/IAO_0000308> .
_:fig1 rdfs:label "example_figure1.svg" .
_:fig1 <http://purl.obolibrary.org/obo/BFO_0000050> oup:pattern_000001 .
```

Listing 1: example triples defining an individual 'ontology use pattern specification,' and specifying the figure it has as its part

```
# a person
_:person1 rdf:type <http://purl.obolibrary.org/obo/NCBITaxon_9606> .
# some HPV inhering in the person
_:hpv1 rdf:type <http://purl.obolibrary.org/obo/DOID_11166> .
_:hpv1 <http://purl.obolibrary.org/obo/RO_0000052> _:person1 .
```

Listing 2: RDF triples defining part of an ontology use pattern for HPV diagnoses

truncate them for display purposes within the system.

In addition to providing a tidy way to keep separate the RDF definitions of patterns in our repository, using named graphs for each pattern also allows us to specify additional information about the pattern, including textual descriptions that explain the intended use, figures that show the pattern rendered in visual format, and additional "metadata" such as the creator, the license for use, etc. This is achieved by using as the name for each named graph a URI that is asserted to be an instance of the 'ontology usage pattern specification' class described above, and inserting the triples that define the RDF pattern within that named graph.

Assertions *about* a pattern instance are made within the main graph of the triple store (that is, outside of any named graph). For example, the RDF triples in **Listing 1** below are used to store the information that the individual `oup:pattern_000001` is an instance of 'ontology use pattern specification,' that there is an individual (here identified by the blank node `_:fig1`) that is an instance of IAO: 'figure' (`IAO_0000308`), that this figure has the filename '`example_figure1.svg`', and that the figure is a part of the pattern specification.

Pattern definition triples

The most crucial piece of a pattern is the set of RDF triples that define the pattern itself. As mentioned above, the system expects blank node identifiers (e.g. `_:person1`) to be used for the instances in these patterns. For example, the following shows triples (part of the pattern shown in **Fig. 1**) representing a person and an instance of HPV that inheres in that person.

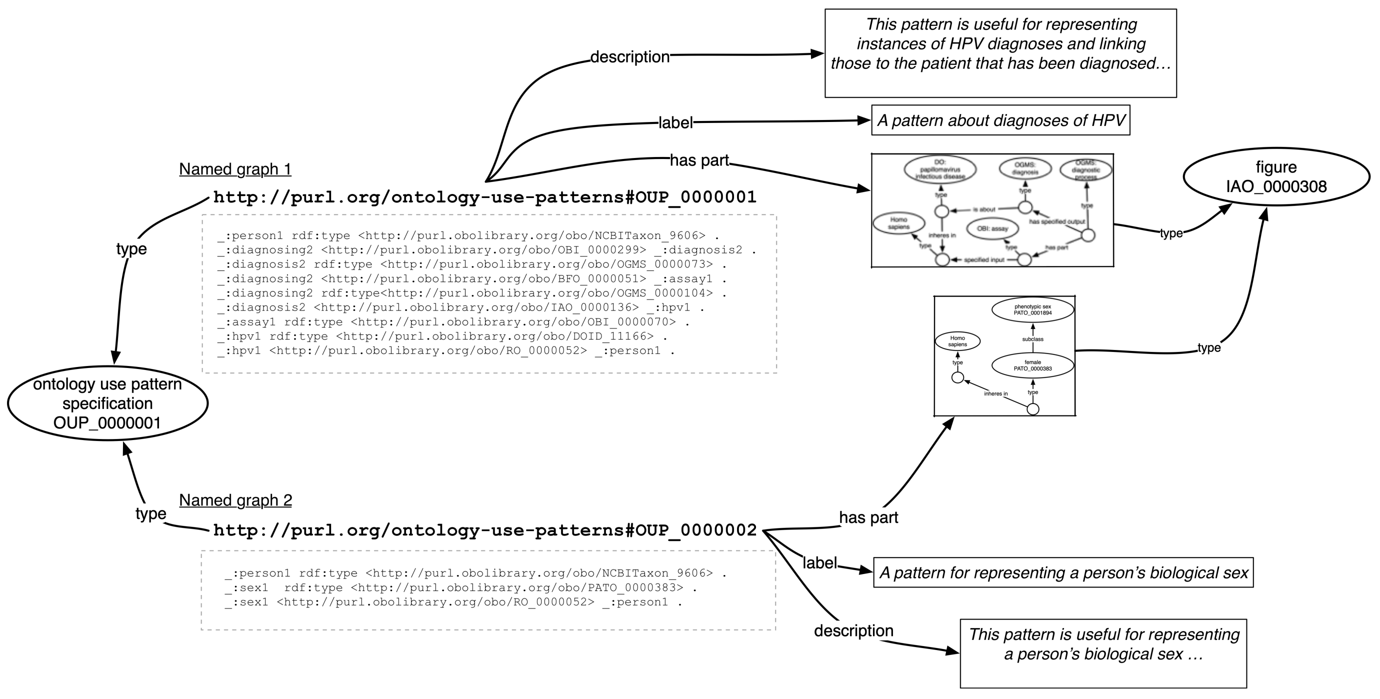
When a pattern definition is inserted into the database, GraphDB replaces its blank node identifier with generated unique blank node identifiers that contain the user's original identifier as a suffix (e.g. `_:person1` becomes `_:genid-bc43f3ab4ec54362af7ed97c9dddcf44-person1`). When displaying a pattern for view or download, the patterns repository interface strips out the generated part of the identifier. As currently implemented this feature does rely on GraphDB's unique way of handling blank nodes internally, and would not generalize to other triple store implementations. Future versions of our tool will implement a more general approach for assigning meaningful names to variables in patterns, for instance by using an annotation property.

Currently adding a pattern definition to the repository through our application involves producing a representation of the pattern as a set of RDF triples expressed in text formatted as N-Triples (12). Possible future work includes a more user-friendly interface for creating such patterns, as discussed more below.

Fig. 4 illustrates the use of named graphs to represent instances of 'ontology use pattern specification', RDF triple patterns within the named graphs, and information asserted about the patterns (descriptions, and other parts, such as figures) in the triple store.

Results

We have created an initial implementation of the ontology use patterns repository proposed and described above. This repository is available at `http://purl.org/ontology-use-patterns`. It is currently populated with several ontology usage patterns used in projects within our group.


description
This pattern is useful for representing instances of HPV diagnoses and linking those to the patient that has been diagnosed…
label
A pattern about diagnoses of HPV
has part
figure IAO_0000308
type
Named graph 1
http://purl.org/ontology-use-patterns#OUP_0000001
type
ontology use pattern specification OUP_0000001
type
Named graph 2
http://purl.org/ontology-use-patterns#OUP_0000002
has part
label
A pattern for representing a person's biological sex
description
This pattern is useful for representing a person's biological sex …

```
_:person1 rdf:type <http://purl.obolibrary.org/obo/NCBITaxon_9606> .
_:diagnosing2 <http://purl.obolibrary.org/obo/OBI_0000299> _:diagnosis2 .
_:diagnosis2 rdf:type <http://purl.obolibrary.org/obo/OGMS_0000073> .
_:diagnosing2 <http://purl.obolibrary.org/obo/BFO_0000051> _:assay1 .
_:diagnosing2 rdf:type<http://purl.obolibrary.org/obo/OGMS_0000104> .
_:diagnosis2 <http://purl.obolibrary.org/obo/IAO_0000136> _:hpv1 .
_:assay1 rdf:type <http://purl.obolibrary.org/obo/OBI_0000070> .
_:hpv1 rdf:type <http://purl.obolibrary.org/obo/DOID_11166> .
_:hpv1 <http://purl.obolibrary.org/obo/RO_0000052> _:person1 .
```

```
_:person1 rdf:type <http://purl.obolibrary.org/obo/NCBITaxon_9606> .
_:sex1  rdf:type <http://purl.obolibrary.org/obo/PATO_0000383> .
_:sex1 <http://purl.obolibrary.org/obo/RO_0000052> _:person1 .
```

Figure 4: Ontology use pattern specification instances represented as named graphs containing pattern RDF triples

The repository system implements a core set of features, including the ability to capture and store information about ontology use patterns within a semantic database. This information about patterns includes:

- RDF definitions of the pattern specifications themselves
- Pattern specification names and descriptions and other metadata
- Figures depicting the patterns

The system also provides the ability to search for and view patterns based on their names and descriptions. We will shortly add the option to also search based ontology terms that are used within the patterns, including text-based search for patterns over term labels and other annotations used in the patterns. This allows the user to identify patterns that use terms of interest by inputting a search string that the system then uses to identify all terms that appear across the entire database whose labels contain the string, determining which named graphs contain triples using any of those terms, and return the list of those pattern specifications for the user to browse.

Once the user has found and loaded a pattern of interest, the system displays the pattern's details in a single page that includes the name, description, and metadata about the pattern; a linked rendering of the pattern's RDF triples representation with ontology term identifiers appearing as clickable links that resolve to the terms themselves via Ontobee; a listing of labels for the terms in the pattern; one or more figures depicting the pattern visually; and a link to download the pattern as an RDF file in the N-Triples format.

Discussion & Future Work

This paper has proposed and presented a solution for the problem of sharing and reusing information about how ontology terms are typically combined into patterns used to instantiate instance data: a repository of ontology use patterns that allows users to create, view, and reuse these patterns and their descriptions. We have implemented an initial release of such a tool and populated it with a diverse set of example patterns related to our work. Development is ongoing to add new features and other improvements.

One planned feature is a diagramming tool for creating RDF instance diagrams with a consistent style like that used in many of the figures in this document, possibly following conventions established by VOWL (13). This tool will then automatically generate the RDF definition of the pattern based on the user's interaction with the diagramming tool. This will allow users to create ontology use patterns within the repository in a single step without going to the separate effort of sketching a diagram and manually creating the RDF pattern, as is currently required. It will also help to ensure the consistency of the main pattern figures used in the repository, as well as the ease of interpreting them. The CAFE project already includes a tool for generating diagrams from its internal instantiation patterns.

Another planned feature is tooling to support copying and editing an existing pattern from within the system itself. We are also considering adding a pattern-based search interface that would allow for more complex queries than the current text-based interface supports.

In addition to this search capability we also plan a more term-based navigation option that will allow users to explore available patterns based on which terms appear in them. In the simplest case, this would involve renderug a page for each ontology term that is used in any pattern within the repository that links to those patterns that use it. In a pattern repository actively populated and used by the biomedical ontologies community, such a term landing page could provide useful information about how, and how often

Acknowledgements

Work on this project has been funded in part with federal funds from the National Cancer Institute, National Institutes of Health under Contract No. HHSN261200800001E. The content of this publication does not necessarily reflect the views or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government. Under this contract the University of Arkansas is funded by Leidos Biomedical Research subcontract 16X011. Funding was also provided by U24CA215109

The CAFE project described in this paper is funded by the National Institute of General Medical Sciences of the National Institutes of Health under award number R01GM111324.

The DCOC work described in this paper was funded by grant number U24OD024957 from the National Institutes of Health Office of the Director through the ECHO program.

The work on representing prescription data described in this paper was supported by the Translational Research Institute (TRI), grant **UL1TR000039** through the National Center for Advancing Translational Sciences of the National Institutes of Health (NIH). The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. This work was also supported in part by BJA-2018-13607: CATEGORY 5 HAROLD ROGERS PRESCRIPTION DRUG MONITORING PROGRAM (PDMP) IMPLEMENTATION AND ENHANCEMENT PROJECTS

References

1. Utecht J, Judkins J, Otte JN, Colvin T, Rogers N, Rose R, et al. OOSTT: a Resource for Analyzing the Organizational Structures of Trauma Centers and Trauma Systems. CEUR Workshop Proc [Internet]. 2016 Aug [cited 2019 Apr 17];1747. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5312685/

2. Jonathan P. Bona, Tracy S. Nolan. Ontology-Enhanced Representations of Non-image Data in The Cancer Imaging Archive. In: Proceedings of the International Conference on Biological Ontology 2018. CEUR-WS.org; 2018.

3. Ceusters W, Smith B. Strategies for referent tracking in electronic health records. Journal of Biomedical Informatics. 2006 Jun 1;39(3):362–78.

4. Ceusters W, Smith B. Aboutness: Towards Foundations for the Information Artifact Ontology. In: Proceedings of the Sixth International Conference on Biomedical Ontology (ICBO). CEUR vol. 1515; 2015. p. 1–5.

5. Gangemi A, Presutti V. Ontology Design Patterns. In: Staab S, Studer R, editors. Handbook on Ontologies [Internet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009 [cited 2019 Apr 15]. p. 221–43. (International Handbooks on Information Systems). Available from: https://doi.org/10.1007/978-3-540-92673-3_10

6. Ławrynowicz A, Potoniec J, Robaczyk M, Tudorache T. Discovery of Emerging Design Patterns in Ontologies Using Tree Mining. Semant Web. 2018;9(4):517–44.

7. Daga E, Presutti V. http://ontologydesignpatterns.org [ODP]. Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008). 2008;401:2.

8. Grinberg M. Flask Web Development: Developing Web Applications with Python. Sebastopol, CA: O'Reilly Media; 2014. 258 p.

9. Ontotext GraphDB^{TM} - a Semantic Graph Database Free Download [Internet]. Ontotext. [cited 2019 Apr 17]. Available from: https://www.ontotext.com/products/graphdb/

10. Gandon F, Corby O. Name That Graph [Internet]. W3C. 2009 [cited 2019 Apr 16]. Available from: https://www.w3.org/2009/12/rdf-ws/papers/ws06/

11. Zhao J, Miles A, Klyne G, Shotton D. Linked data and provenance in biological data webs. Brief Bioinform. 2009 Mar 1;10(2):139–52.

12. RDF 1.1 N-Triples [Internet]. [cited 2019 Apr 17]. Available from: https://www.w3.org/TR/n-triples/

13. Visualizing Ontologies with VOWL | www.semantic-web-journal.net [Internet]. [cited 2019 Apr 17]. Available from: http://www.semantic-web-journal.net/content/visualizing-ontologies-vowl-0