Helping Scientific Researchers Make Better Use of Ontologies
Tools, Methods and Best Practices

Gary Berg-Cross and James A. Overton

Abstract

The idea of using ontologies and other related/semantic resources to improve computing has long existed, with many initiatives to develop methods and tools that could be widely employed by the scientific community. While the effort to develop and use ontologies has matured, many barriers remain to help typical end-user researchers develop and employ ontologies. These include questions related to how to initiate the development of an ontology, what tools to use, and how to maintain an ontology after its development. In addition, it is difficult for domain users and data specialists to express and maintain knowledge represented in formal languages such as first order logic and or OWL for ontologies. Motivated in part by the influence of big data and the widespread adoption of FAIR principles that encourage the use of formal semantics, this communiqué provides an overview of the current environment of ontological methods and the tools and best practices across the ontology development lifecycle. The communiqué concludes with a list of current challenges that require continued research. While the emphasis is on bio-ontology tools, ontology technologies can be applied to other disciplines as diverse as the Earth sciences and finance, or have more general use such as in knowledge graphs.

Introduction

Early ontology summits, such as the Ontology Summit 2007 "Ontology, Taxonomy, Folksonomy: Understanding the Distinctions", have attempted to explain the value of ontologies and related semantic resources. The have also provided guidance on usage as found, for example, in the Ontology Usage Framework (2011) co-championed by Michael Grüninger, Michael Uschold, and Nicola Guarino. This guidance is also useful for other artifacts such as terminologies, thesauri and vocabularies, collectively known as knowledge organization systems, that can be used to annotate data. Even these early efforts on ontology development made clear that building, maintaining, and using an ontology is an iterative process that often calls for ongoing collaboration and communication between stakeholders, domain experts, and ontology developers. With Big Data, open and

well managed community ontologies are increasingly important for open science (Bennett and Baclawski, 2017; Musen et al., 2018).

There is now a growing commitment to data sharing from scientific researchers and universities (Sorbonne Declaration, 2020). This commitment has been codified in the Findable, Accessible, Interoperable, and Reusable (FAIR) guidelines (Wilkinson et al., 2016). Ontologies and semantic approaches can facilitate some of the FAIR guidelines, which emphasize machine-actionability (i.e., the capacity of computational systems to find, access, interoperate, and reuse data with none or minimal human intervention) (FAIR Principles, n.d.). Among other things, a principled use of semantics would facilitate the creation of metadata documentation needed to support data repositories. Much day-to-day work can involve working with data scientists to make the data FAIR and address the challenges of many formats and using ontologies. How do we make the concepts easier to use? Too often, some of the semantic aspects of FAIR are particularly difficult for scientists. Furthermore, the idea of extending FAIRness to ontologies would obviously be useful, but the challenge is how to do so. Issues of grounding ontologies into deeper semantics raise foundational issues. The work of the OBO Foundry (OBO Foundry, 2023), for example, demonstrates how "a sizable, federated community can be organized and evaluated on objective criteria that help improve overall quality and interoperability, which is vital for the sustenance of the OBO project and towards the overall goals of making data FAIR." (Jackson et al., 2021).

We begin this communiqué with a section on defining the problems that are encountered by scientists when confronted with assuring compliance with the enhanced semantics requirements of the FAIR guidelines, particularly with respect to ontologies and other semantic resources. In practice, the actual development, maintenance, coordination, and use of ontologies and other semantic resources in science remains complex because of the reliance on a sophisticated interconnected web of tools, methodologies, standards, various knowledge artifacts and community practices, at the intersection of science, software development, and information management. While the semantic community increasingly understands how to address some of these challenges, the larger community of scientists and researchers often do not.

The main sections of communiqué give an introduction and overview to the many methods, tools and resources that have been developed by the semantic community. Specifically, the sections are:

Methodologies We start with the basics of ontology development (Noy and McGuinness, 2001; Noy et al., 2010) and show how basic engineering has advanced over time to be more socio-technical in nature (recognizing the impact on people as well as the technical aspect).

Resources As with software development in general, one starting point for ontology development is to reuse existing ontologies.

Techniques A large variety of techniques are available for ontology design tasks, including customizable forms such as design patterns, templates, modular modeling, as well as upper level domain ontologies.

Tools There are many tools for managing the later phases of ontology development as well as ontology maintenance and use. This section describes a selection of such tools.

Knowledge Graphs A knowledge graph (KG) is both a knowledge representation language and a knowledge system that helps to organize and structure data and information in a way that is can be easy to understand, search and navigate. Mapping ontologies to KGs can help ensure that the both data and ontologies satisfy the FAIR guidelines.

A simple, running example illustrates the nature of the work. Note that only a sample of methodologies, tools, and models (e.g., ontologies) are provided, such as those mentioned in the Summit presentations. The reader is encouraged to research the broader global content on these topics.

The communiqué ends with conclusions and recommendations for continued research efforts. The appendices provide a glossary of important terminology and scenarios. Our goal is to help scientific researchers and other practitioners to take advantage of the many tools and techniques that have been developed by the ontology community to make better use of ontologies.

Defining the Problem

There have been many discussions in online forums and academic conferences on the challenges of ontologies. Interested users or developers often voice concerns about problems and obstacles preventing them from achieving a satisfactory level of competency, particularly on semantic topics. A recent Earth Science Information Partners (ESIP) workshop "Semantics Adoption in the Earth and Environmental Sciences: Successes and Roadblocks" organized in January 2023 by the ESIP Semantic Harmonization co-leads Gary Berg-Cross and Ruth Duerr, identified several issues that were preventing people from getting started with, or using, semantics more in their work. These included technical issues such as difficulties with formal logical languages. Some of the activities do not make sense to non-ontologists.[1] Others are caused by starting points with captured knowledge that is too informal, may allow duplicates in tools like spreadsheets (an issue for populating knowledge graphs for example), allows ad hoc creation of classes with no thought of referential integrity (Alani, Harith et al., 2002) or allowing a wide variety of schemas to express the same or similar knowledge. Each of these makes FAIR interoperability of data resources difficult. Similar aspects were the subject of last year's summit on Disasters (Sharma et al., 2022).

Domain size and (in the context of manual development tasks) the repetitive nature of adding entities are also well-known challenges. It is important that an ontology supports a large number of classes, many of which may have similar structure so basic ontology editing tools such as Protégé can frustrate users. Repetitive tasks are often painstaking, and spreading editing over multiple people can introduce errors. Sometimes collaboration makes editing worse (Skjæveland, 2023).

There are only a few examples of mature technological support for most ontology engineering methodologies. Generally many tools are available, but often are: too technical and low-level for non-programmer use; not mature; not widely adopted; or poorly integrated with other ontology tools or with common Integrated Development Environment (IDE) envi-

[1] The ontology community published recommendations for the body of knowledge that an ontologist should learn and the skills that an ontologies should acquire in Neuhaus and Smith (2010) and Neuhaus et al. (2011).

ronments.

Overall, IDE tooling is currently insufficient for ontology development. The development phase of ontologies requires better approaches than currently available for visualization of models and help in conceptual modeling, a challenge long recognized in knowledge engineering (Katifori et al., 2007). To some the importance of a conceptual solution, and the lack of adequate diagrams is more of a problem than challenges of OWL representation or exact serialization. Semantic solutions (Fürber and Fürber, 2016) must be human-understandable, requiring the need to handle and hide complexity using methods and tools that maintain full logical axiomatization "in the background." The Open Ontology Repository concept was discussed in the Ontology Summit 2008 and requires infrastructure support for the sustainable repositories of interoperable or stand alone mature ontologies (Obrst and Musen, 2008).

Problem size is evidenced in the comment at the ESIP discussion of semantic issues, "Designing a query to find ALL chemical participants in a biological process is too hard." In some cases, there are misconceptions about semantics itself or its role in supporting data management, metadata tagging, and data interoperability. Examples of misunderstanding include: "There needs to be one ontology." or "Semantics is not robust enough to describe data to the level needed for data integration/harmonization" to "I can find ontologies online, but how do I know whether they are endorsed by the scientific community?" or "The real word is fuzzy, semantic systems are primarily discrete." Other types of problems deal with programmatics or are organizational such as "How do I start (a project)?" or "Our funding models are broken." Still others were "I am not sure where to start" or expressed practical needs for, "Practical applications bringing all this work to bear on discovery for busy scientists" or "if I search on a term it may appear in a couple places in the ontology." There are simple complaints such as "How do I make sure that whenever I make a change, I didn't break anything?" or "I feel like I have to create rules for how to choose between different ontologies. Lots of work." Interoperability and ease of use appear to be the needed facilitators for easy adoption among others.

A common problem concerns how to start the development of an ontology. The experience of the OBO Foundry is that it is difficult to

attract and keep end-users productively involved, which is important for building and maintaining large ontologies.

Once an ontology development project has started, technical issues arise including language difficulties, consistent modeling and how to collaborate as well as automated support (Skjæveland, 2023).

Knowledge representation issues include questions on how to simplify OWL formalization. The well known Manchester syntax for OWL 2 ontologies represents coding at a too low level and can be difficult. Also challenging are tools that support work on conceptual modeling that will later form the basis of an ontology.

Methodologies

A simple, but well accepted, view of the ontology engineering process (Figure 1) was developed in 2001 and begins by determining the scope and uses of an ontology (Noy and McGuinness, 2001). This is followed by conceptualizing the ingredients of the ontology, including: identifying and enumerating the classes and properties that will be used to represent domain concepts and relationships of interest, considering relations that can be axiomatized as rules that will govern the logical behavior of the ontology. A formalization step using low level tools, such as Protégé, which is often difficult for domain researchers, follows conceptualization and involves selecting a formal language that will be used to logically represent the ontology. This may organize the classes into a hierarchy with subclasses, while also specifying their properties and interrelationships in a machine-readable format. Finally, the classes are populated with instances, yielding an ontology that is sometimes described as a knowledge base.

Proposed requirements for ontologies have grown over the last 20 years and now include more complex methods for things such as the uses of glossaries and templates, and the iterative maintenance of large, interrelated ontologies. This reflects the need to support:

- A large number of classes, potentially requiring collaborative community conversation. For example, there are millions of classes across more than two hundred OBO Foundry ontologies.
- A large number of standards and related semantic resources such as

Figure 1: Classic ontology development lifecycle (Noy and McGuinness, 2001)

glossaries. When compared, they may have apparently incompatible conceptualizations within the specified scope or universe of discourse.

- Semantically consistent modeling. There is also a need for a modeling that works for the same category of things in the same way, across different standards.
- Community collaboration with different levels of experience and interest. This can introduce problems. Some support is needed to make collaborative development easier.
- Contributions from varying backgrounds and competencies (technical and ontological), reflecting the fact that community models should be consistent across all contributors!
- Mechanisms to make the development process easier; for example, automated mechanisms for quality assessment and verification.

Taken together, these requirements reflect recent methodologies with a community focus and greater level of detail, and are worth considering to frame an understanding of modern ontological engineering. In contrast to earlier methods, approaches such as the NeOn methodology (Suárez-Figueroa et al., 2015) provide a richer lifecycle methodology founded on the four pillars listed below. It is useful for discussing a range of practices within ontological engineering phases. The NeOn Toolkit includes several Protégé plug-ins to aid the various activities in the ontology engineering process.

1. A glossary for activities in the methodology. This aids a user in understanding the terms used. Appendix 1 is an example of a glossary.
2. A set of scenarios for building ontologies and ontology networks. Appendix 2 is an example of a set of scenarios and a supporting figure.
3. A waterfall, but also an iterative ontology life-cycle model. An iterative life-cycle may be seen to support ontologies whose scope helps to represent a dynamic domain (e.g, healthcare) which is modeled over time.
4. A set of prescriptive methodological guidelines for performing specific activities. These guidelines can aid in obtaining and re-using terms from external ontologies.

Ideas on how to engineer ontologies go well beyond technical elements are increasingly socio-technical in that the overall process requires ongoing collaboration and tool-use as well as communication between stakeholders, domain experts, and developers.

The following steps are socio-technical best practices that can help avoid commonly occurring ontological errors and can help format and refine domain terminology (Rudnicki, Smith, Malyuta, and Mandrick, 2016):

1. Form a team. Ensure your team includes both subject matter experts familiar with both subject-matter (the entities in your domain) and data (the information resources that the ontology will be used to integrate and analyze and to make discoverable).
2. Ensure that your team includes persons with ontology-building experience. (Facility with ontology software does not imply the ability to create ontologies.)
3. Identify the primary tasks your ontology will be designed to realize.
4. Identify the domain of your ontology – the types of objects and attributes and processes which the ontology will need to represent.
5. Identify the primary bodies of data your ontology will be used to annotate.
6. Be aware that your goal is to maximize the ability of your ontology to address these primary tasks but without detriment to its ability to address secondary uses not yet identified. (Experience shows that

secondary uses are often significantly more important than primary uses, and that secondary uses are almost always what guarantees the enduring value of an ontology.)[2]

In NeOn, the requirements phase and team-building activities are followed by a phase in which reuse is considered. This phrase may include a search and evaluation of non-ontological resources (NOR), such as glossaries with terms defined for human understanding but not axiomatized for automated processing. Proper selection of these resources may help ensure grounding in domain concepts already used by a community as part of communication. Note, however, that the reuse of either (non)ontological resources does not in itself ensure consistency or implementation of the intended semantics (Rovetto, 2023).

Several recent developments help illustrate the value of the second NeOn scenario in Appendix 2, Reusing and re-engineering non-ontological resources. Indeed, since there are many, diverse sources of data in different formats on the web having a standard source is useful. Wikidata, for example, is one that is a free and open non-ontological graph knowledge base with descriptions and qualified statements about uniquely identified entities and their properties such as people, places, things, and events in a similar form to RDF. The Wikidata graph is easily queried about data.

Although it does not include strict axiomatization and its knowledge may be shallow, Wikidata's concepts and relationships can be a useful source of ideas for ontology development. Since properties in Wikidata are community created, however, there is often no direct mapping to an ontology's property relation. Mapping to support interoperability between semantic resources is complex, but is aided by the use of Internationalized Resource Identifier (IRI) for identity and basic semantics. For example, there are some mappings of Wikidata terms to OBO Foundry ontologies including reuse of relationships based on a mix and match approach to aligning two independently developed semantic resources. Simple Knowledge Organization System (SKOS) relations such as "close match" can be

[2]The main strategy to ensure future-proofing against problems in addressing secondary uses is to ensure that the terms and definitions in your ontology are of broad understandability and validity (rather than being understandable and valid only by your immediate collaborators and only when used in relation to your currently available data).

used to document relations (Hoyt, Hoyt, and Gyori, 2023).

Science on Schema.org is another non-ontological resource designed to help scientists make their data more discoverable and accessible to others. To do this, Science on Schema.org provides common publishing patterns for describing research data guidelines about scientific datasets and resources using an extended Schema.org vocabulary. Essentially, it is a metadata schema with an extended vocabulary to consistently describe data and related material such as temporal and spatial coverage, or people's roles in data. Science on Schema.org takes a step toward interoperability between vocabularies which in turn may be a resource for the ontology population.

Guidelines for starting with NOR cover a wide range of topics, including:

- How to describe the different types of scientific data
- How to provide links to related resources
- How to use Schema.org properties to describe the data

An important benefit of using Science on Schema.org is increased data interoperability and reusability with modest formal semantics that can be further aligned to ontologies as needed. Domain work between projects runs into interoperability challenges across ontologies that need to be monitored, including the variation in terms and relations, metadata documentation, how responses are managed to make changes and how to control the chaos in ontology prefixes.

Data models themselves can be a useful non-ontological sources of information. The Linked Data Modeling Language is a data modeling framework that can be used to describe many kinds of data models: from value sets and flat, checklist-style standards to complex normalized data structures that use polymorphism and inheritance (LinkML, n.d.). LinkML is designed so that software engineers and subject matter experts can communicate effectively in the same language, while also providing the semantic underpinnings to make data conforming to LinkML schemas easier to understand and reuse computationally in the semantic web. The LinkML metamodel provides the ability to map a model class or attribute to an existing ontology class (via SKOS exact, narrow, related, etc. mappings), to a

set of ontology classes (via semantic enumerations), and/or to declare that the URI for a model class or attribute is exactly the URI for an existing ontology class.

Additionally, the LinkML framework includes tools to serialize data models in many formats including, but not limited to: JSONSchema, OWL, RDF, SQL-DDL, and Python Pydantic classes. It also includes tools to help convert both instance and class data from one model serialization (like YAML) to a different model serializations (such as OWL). LinkML provides validation software at both the instance and schema level, software to navigate and query model metadata (via LinkML schema view), and tools to bootstrap a LinkML schema from another framework (LinkML schema automator). In addition, it can auto-generate documentation and schema diagrams.

Other ontology development methodologies, or approaches for development methods, include Agile methods, and the Methontology methodology which may be a precursor version of NeON.

Resources

All of the ontology engineering methods begin with either reuse of existing ontologies or bespoke development of new ontologies. If the starting point is to reuse existing ontologies, then candidate ontologies are analyzed and evaluated for suitability, as discussed in NeOn Scenario 3. Choices must be made because a given domain or topic can be modeled in various ways, resulting in distinct yet equally-valid ontologies. Candidate ontologies may be open-source or proprietary and available for purchase. Identifying the appropriate ontology for a specific task can be challenging, especially for those who are not familiar with the task; however, ontology repositories now exist with searchable collections. Some examples include the following.

- Linked Open Vocabularies (LOV) provides a curated collection of vocabularies, broadly construed to include ontologies (Linked Open Vocabularies, n.d.)
- The OntoPortal application provides a means for creating repositories or libraries. BioPortal for example, has various biological and medical ontologies (Bioportal, n.d.).

- The OBO Foundry, a library of biomedical ontologies, covers a wide range of topics, including from Biology (genes, proteins, cells, tissues, organs, and organisms) to Biomedicine covering diseases, symptoms, treatments, and procedures; to chemicals, compounds, and reactions; ecological information about for organisms, habitats, and ecosystems. However, the requirements for being included in this library may be more restrictive or otherwise undesirable for some. In any (for any library or repository), requirements or criteria for inclusion should be evaluated and clearly understood. Note in passing that the Ontology Development Kit (ODK), developed by the OBO community, and covered in more detail in the Tools section below, is one tool that may be used to reuse an existing ontology and/or its parts.
- Another example of a library for reuse is MODL, a curated Modular Ontology Design Library (Shimizu, Hammar, and Hitzler, 2021). It has collected well-documented ontology design patterns, drawn from a wide variety of interdisciplinary use-cases. MODL contains over 100 ontology design patterns (ODPs), organized into several categories include basic patterns similar to OBO's templated OWL patterns such as metapatterns for the organization of data; space and time events; the movement; agents and roles; along with descriptive details such as quantities and units, partonymy/meronymy, provenance and identifiers. More on the role of ODPs is discussed in the Customizable Forms section below.

Techniques

The interoperability of data via ontologies is a key aspect of FAIR. However, ontologies developed for different domains also need to be interoperable. The biomedicine community is a good example of a community dealing with both data and ontological interoperability in a principled and practical way. The OBO Foundry's strategy has four levels. At the base level are things like common FAIR principles that allow ontologies to be found, accessed, reused supported by FAIR criteria such as IRIs and metadata that will allow a degree of interoperability. This is further supported at the next level up by agreement on common formats and the use of integrated tools, such as OWL tools like Protégé, and OWL API as well as higher-level development and management tools such as ROBOT en-

abling the ability to read and write ontologies in commonly understood formats. Level 3 focuses on community developed semantic resources such as shared vocabularies and upper level reference models that support standardization and integration between ontology modules. At the top (Level 4) are shared, general design patterns for building domain ontologies such as dead simple OWL design patterns (DOSOP, 2023) which are a simple templating system for documenting and generating new OWL classes. The templates themselves are designed to be human readable and easy to author. Separate tables (TSV files) are used to specify individual classes. ROBOT templates, and the Reasonable Ontology Templates (OTTR) are other templates that serve to develop general modeling patterns that can be used to start an ontology and share them across a community. In the following subsections, we discuss simplifying tools and techniques from the various levels as well as how they work together.

Customizable Forms

Ontology design templates (ODTs) and ontology design patterns (ODPs) are time-saving tools for an ontology designer by providing easily understood starting points for building a new ontology. The modular parts are designed to interoperate, thus improving the efficiency of the development process (Shimizu, C. Hitzler, P and Krisnadhi, 2020). ODTs and ODPs capture human conceptualization using a machine parseable constraint language (Hitzler and Shimizu, 2018). When selecting relevant templated patterns there are trade-offs to consider between generality and specificity. Abstract modeling patterns may provide a general structure for an ontology, as well as some of the basic concepts and relationships that are likely to be needed before use of more specific patterns. Together, they represent a useful middle ground between reusing a complete ontology and making a new one from scratch (Blomqvist et al., 2016).

OBO Patterns

The OBO Foundry encourages the adoption of a curated collection of rigorously defined and extensively validated OWL design patterns. These patterns, formulated by domain experts, bridge the gap between human conceptualization and the corresponding, known data instances. Frequently occurring patterns represent a common starting point that may help with consistency and interoperability between ontologies. The ODP schema

can impose structure on data, encoding relational knowledge so data may interoperate with other data that has been built from the same pattern.

Many OBO projects provide templates that allow for efficient sharing and reuse among developers, promoting collaborative efforts and facilitating ontology updating and maintenance. By implementing these templates, developers can construct ontologies with coherent semantics, seamless interoperability, and user-friendly accessibility.

More Complex Patterns

Some templated patterns cover and connect a larger number of concepts. An example is the Recurrent Event Series ODP that has 4 main concepts (Carriero et al., 2019):

1. Recurrent Event Series - A recurrent event series is a collection of events that recur at regular intervals.
2. Event - An event is a single occurrence that happens at a specific time and place.
3. Unifying Factor - A unifying factor is a property/characteristic that is shared by all the events in a recurrent event series.
4. Recurrence - Recurrence is the property of an event or event series that occurs at regular intervals.

Reasonable Ontology Templates

The templating language and toolkit used in OTTR is another example of how to simplify the process of structurally coding knowledge such as classes, properties, and individuals so they can be expressed in OWL (Skjæveland et al., 2018). These templates also represent the initial, general building blocks of the ontology and serve as the starting point for further refinement. Taken as a whole the OTTR approach helps:

- Avoid unnecessary repetition
- Encapsulate complexity
- Support uniform modeling
- Separate design and content

- Ensure input completeness via type checking and consistent use of IRIs
- Simplify input format (using parameters, expandable lists and macros)
- Support parameterized substitution and provides macro expansion similar to macro systems found in programming languages, where macros are used to define reusable code fragments that can be expanded into larger expressions or statements.

More detail on OTTRs and associated methods are summarized in Appendix 3.

Modular Ontology Modeling

The domain reference ontology discussed in more detail in the next subsection typically are composites that employ one or more patterns. These reflect integration of smaller modules and intended to be reusable as an artifact and not just a template (Hahmann, 2023). Indeed, modular ontology methods or "pattern-mediated methods" built around templated ODTs and ODPs represent a systematic, alternate approach for building up ontologies from meaningful smaller parts.

The idea is to start bottom-up from concepts which are then grouped together into larger modules. This views an ontology as made up of smaller, more manageable modules that are connected using special, compositional semantic relations. ODP structure also called "metadata scaffolding" can be used to represent background knowledge, improve alignment and provide a mechanism for improved reusability (Shimizu, Hammar, and Hitzler, 2023).

Modular solutions use tools, such as the previously discussed templates that understand key aspects of the ontology language (OWL, RDF, or RDFS). The Comprehensive Modular Ontology IDE (CoModIDE) is a plugin for Protégé that allows bringing together such modules produced in a 9 steps method listed below (Shimizu, Hammar, and Hitzler, 2019):

1. Define the use case
2. Write up competency questions
3. Identify key notions

4. Match patterns to key notions
5. Template-based instantiate the patterns
6. Systematic axiomatization
7. Assemble the modules
8. Review final product
9. Produce the OWL artifacts

An attraction to this method like templates themselves is that, in theory, ODPs provide a start on work for projects that lack some ontological expertise. The intent is to enable domain experts to reuse existing and tested best practices for design decisions. However, in practice the adaptation of ODPs as tools for ontology engineering by domain experts remains slow. (Krieg-Brückner, Mossakowski, and Codescu, 2021).

Upper Level Domain Models

Another starting point for ontology development is an upper level domain model. These models may be understood as broad but relatively domain-specific models. For example, the Core ontology for Biomedicine (COB) is a high-level ontology describing the Biomedical domain. It is motivated in part by a need for a simplified, user-friendly layer above application oriented ontologies at an intermediate level, but below a more abstract, general top level ontology such as the Basic Formal Ontology (BFO) as shown in Figure 2 (Otte, Beverley, and Ruttenberg, 2021). Some of the ideas supporting the use of a high level reference model include "upper level domain reference models" such as COB in the Biomedical area and HyFo (Hahmann, Stephen, and Brodaric, 2015), a foundation for hydrological flow dynamics, axiomatize deep knowledge of core domain concepts. This provides a solution-independent specification (e.g., a domain conceptualization) using a clear and precise description of domain entities at a level of detail such that other domain ontologies/standards including in applications can be expressed using this terminology. These have proven useful in interdisciplinary (domain, programming, data, metadata, semantic technology and ontological) community efforts.

Practical work needs something closer to a domain to organize concepts. An upper domain model like COB provides a central, consistent place where different ontologies (say within OBO) can discuss and agree

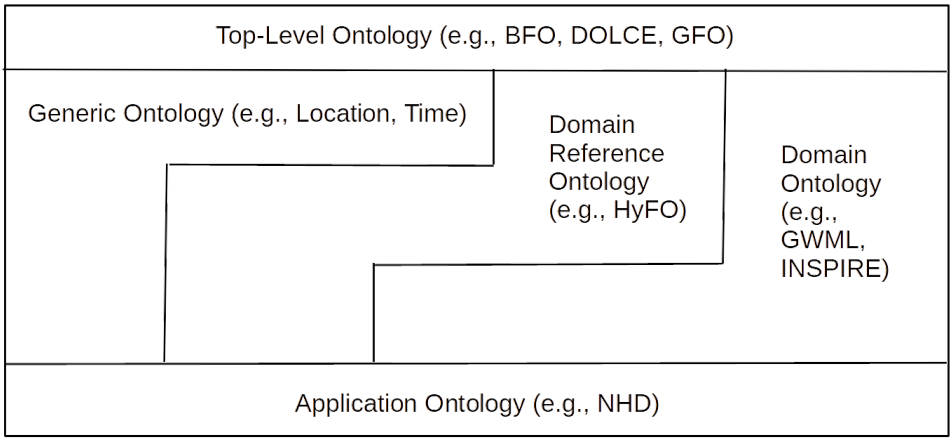

Figure 2: Types of ontologies, indicating where each type occurs within the class hierarchy (based on Hahmann, 2023)

where the different root terms across OBO (like mass) belong in the COB hierarchy. One advantage is that termed classes with the same semantics in different ontologies can be related as "equivalent" and IDs can be swapped out as needed. Another advantage is that COB can be modular, can use agreed-upon standardized relationships, and can have associated tools and community standardized best practice techniques, such as how to use metadata to annotate ontologies to manage it. Some of these tools are discussed the subsection below.

Tools

A major phase of work following reuse is the various conceptual and constructive activities that allow population and later implementation of an ontology in a logical language. This requires careful consideration of the relationships between concepts and entities. The process can be time-consuming and requires some level of expertise. Construction is a collaborative process which can be helped by tooling.

There are many tools to consider for population and later phases of work. For example, the OBO community has supported efforts into making quality ontologies easier to develop and maintain via concentration on practices and tools supporting open collaborative workflows, often

centered on GitHub, including editing, quality control releases and infrastructure. Taken together, the community effort involving professional data and knowledge curators, as well as ontologists help make OBO ontologies more trustworthy and actively supported and responsive to new requirements.

With some ontology edits it is difficult, if not impossible, to model consistently since humans are not good at repetitive tasks. As an aid tools from the OBO community support repetitive tasks are discussed in the subsections below.

ROBOT

ROBOT is one of the tools in the OBO development stack (Jackson et al., 2019). It is a well documented, command-line tool and library for automating OWL ontology development tasks. Based on shared best practices, it uses Java code (e.g., interface command to extract or merge ontologies or core operations such as comparing ontologies) to automate repetitive tasks needed to create and manage OWL-based ontologies such as controlling ROBOT from Python. ROBOT can also be used to:

- Convert an OWL ontology to various formats
- Extract a module from an ontology
- Merge an import to an ontology or unmerge by removing axioms from an ontology
- Mirror by making a local copy of an an import chain
- Report ontology measures and
- Provide control on ROBOT operations via Python scripts.

A ROBOT template can be used to create an ontology or selected parts of an ontology.

The Ontology Development Kit

Constructing an ontology involves developing and implementing the ontology in a computer-readable format. This may involve using an ontology editor such as Protégé or a programming language to create the ontology. Higher-level tools to simplify the process but also provide a richer ontology. The ODK, which has been developed and maintained by the

Open Biological and Biomedical Ontologies (OBO) Foundry, uses a Docker containerization platform to hold a toolbox that can be used to develop and maintain ontologies. Docker makes ODK easy to install and use, as well as being scalable and secure.

The released version of an ontology can take several forms, depending, for example, on whether there has been reasoning over the ontology or whether it contains imported axioms from external ontologies. To facilitate interoperability and modular reuse of ontologies, the ODK defines a few standardized release products, such as the "base" product, which contains only native axioms, and the "full" product, which also includes imported axioms and axioms inferred by logical reasoning (Matentzoglu et al., 2022).

The ODK tools can be used to:

Create new ontologies The ODK provides a set of tools that can be used to create new ontologies from scratch as well as to edit them.

Publish ontologies The ODK provides a set of tools that can be used to publish ontologies to the web.

Test ontologies The ODK provides command-line tools that can be used to test ontologies such as checking for errors and ensuring that the ontology is consistent.

Document ontologies ODK provides command-line tools that can be used to generate documentation for ontologies such as generating HTML documentation and OWL documentation. Note: ontology design pattern approaches also recognize similar documentation requirements for a quality ontology (pattern). These include documenting the pattern with a Schema Diagram, examples of Pattern Instantiation, a list of competency questions, axiomatization documentation, a link to an OWL File, Pointers to Related Patterns, and pattern Metadata.

Ontology Access Kit

The Ontology Access Kit (OAK) is a set of Python libraries and tools supporting access to ontologies. Some ways in which OAK can be used include:

Loading ontologies OAK provides a simple way to load ontologies in various formats, such as OWL, RDF, and OBO. Once an ontology is loaded, it can be queried and manipulated using the OAK API.

Querying ontologies OAK allows people to query ontologies via the OAK API, which provides an easy way to express complex queries over an ontology to find all classes that have a particular property, all individuals that belong to a certain class, or more complex reasoning tasks.

For all these reasons OAK can support evaluation and testing to ensure that an ontology meets the requirements and specifications of the stakeholders and that the ontology is consistent. This may involve testing the ontology against a set of use cases or scenarios.

Dashboards

Once in use, an ontology needs to be maintained and updated to ensure it remains relevant, accurate and useful. Maintaining an ontology involves adding new content (e.g., classes) or revising existing content to reflect changes that occur in the domain of interest. For example a project may want to update and change the way they model a concept like "river" so that in a drought a river need not contain water, but still acts as a container for water. When adding new terms, a merging phase that aligns resources typically takes place. If a modular approach is used, merging is followed by a reengineering phase of the new terms into modules.

The conceptualization of a given concept or class may also change over time. In the context of an open-source project, such as OBO, ontology change starts with submitted change requests via an online platforms (such as GitHub), followed by planning of changes, implementing the change and evaluating the effect to decide if a proposed change should be implemented. With large ontologies, as in the OBO Foundry, the management of a consistent ontology across version updates and their dynamics is a major challenge and requires tool support. If the ontology can be localized to one cultural language, a formal implementation phase to represent the conceptualization in an ontology language is next followed by a Maintenance Phase for improvements and corrections.

The OBO Dashboard has been developed and maintained by the

Open Biological and Biomedical Ontologies (OBO) Foundry. The OBO Foundry provides a set of principles guiding ontology development, offers help in reviewing new ontologies, and aids existing ontologies to improve their community alignment, metadata standards, and provenance annotations (Jackson et al., 2021). The work of the OBO Foundry is also assisted by a tool called the Ontology Quality Assessment Toolkit which automatically assesses all references (e.g., IRIs or CURIEs) in the ontology, including in semantic mappings, provenance, properties, and other metadata for consistency (OQUAT, n.d.). OQUAT produces web-based reports that community members can review and use as the basis for improvements. Typical consistency issues detected by OQUAT include typos within prefixes (e.g., Wikidata instead of wikipedia), non-standard usage of identifiers which violate the expected pattern, or unknown prefixes that aren't already registered in the Bioregistry (Hoyt et al., 2022; Bioregistry, n.d.).

Assessment features included in a Dashboard to gauge the quality of an ontology, include:

Metadata OBO Dashboards provide a summary of the metadata for an ontology, such as the name, version, and license.

Statistics OBO Dashboards provide statistics about the size and structure of an ontology, such as the number of classes, properties, and individuals and feedback to users on updates.

Conformance since interpretation of principles is not always easy, OBO Dashboards provide operationalized definitions so that responses can be used to assess whether an ontology conforms to the OBO Foundry principles, a set of guidelines for developing high-quality ontologies.

ROBOT Report OBO Dashboards can be used to generate a ROBOT Report, which is a comprehensive report that assesses the quality of an ontology against a set of criteria.

Knowledge Graphs

Knowledge graphs are applications that help to organize and structure data and information in a way that is easy to understand and navigate.

Because they have some formal structure that is intuitive to people they can be helpful to humans and processable by machines.

As discussed in Baclawski et al. (2021) ontologies are a semantic resource along with RDF encoded data. They are useful in building KGs by helping to structure knowledge graphs populated with RDF triples and by providing a common consistent vocabulary and set of relationships for describing the data. If done effectively it makes it easier to understand the KGs data, to reason about it, and to integrate it with other knowledge graphs.

However, there is an associated task and challenging task of mapping data in the KG to the appropriate concepts and relationships defined in the ontology. Keeping a good mapping as the KG evolves can be challenging. The experience of the OBO Foundry community's work and the development of Ubergraph is illustrative.

Ubergraph is a recursive RDF hypergraph data structure providing a public SPARQL endpoint to 50+ OBO ontologies loaded and pre-reasoned with simple triples. Ubergraph is used in the NCATS Biomedical Data Translator Program to create a unified view of biomedical data, such as genetic and clinical trial data as part of tracking research results that can improve the discovery and translation of new biomedical knowledge. As opposed to an axiom based representation found in the OBO Foundry ontologies, it is a general-purpose graph data structure for simplified querying supporting directed and undirected edges, weighted edges, node and edge attributes, and a mixture of directed and undirected edges within a single graph.

A knowledge graph outside the BioMedical area of note is the NSF-funded KnowWhereGraph (KWG) which is a densely populated cross-domain knowledge graph that incorporates over 30+ thematic and "placial" datasets into a KG using over 13.5 billion triples. Although some large ontology design patterns like Sensor, Observation, Sample, and Actuator (SOSA) are used along with space and time ontologies, unlike Ubergraph, it's starting point is not a family of related domain ontologies, but data sets that are formalized as concepts - object, data, and annotation properties. The schema strategy is to isolate instantiated patterns into modules, and then interconnect them into a coherent graph.Together, they provide a

wealth of highly diverse sources (hazard events, places, and people) of relevant spatial and non-spatial information about the location of features (like rivers), the relationships between features, and the properties of features to form an open, extensible, standards-based, and spatially-explicit knowledge graph. Together with a geo-enrichment service stack for applications in the environmental domain, the KWG schema integrates different kinds of data and their relationships. Users may not have to have special knowledge such as a special language of space and time to access information about topics such as:

- Environmental monitoring: The KWG can be used to monitor the environment for changes, such as changes in land use, changes in water quality, and changes in the distribution of species.
- Natural disaster response: The KWG can be used to support the response to natural disasters, such as floods, earthquakes, and wildfires.
- Sustainability planning: The KWG can be used to support the planning of sustainable development projects, such as the development of renewable energy projects and the protection of natural habitats (Janowicz et al., 2022).

The knowledge engineering for KWG uses the "pattern-mediated methods" discussed in the Modular Ontology Modeling section above combined with metadata scaffolding to quickly assemble background knowledge, improved alignment and an overall schema to improve reusability. OpaL, (Object, Process, Actor modeling Language) is an ODP representation language that can be used for assembling modules. It is designed to support domain experts who need to build an ontology by providing a limited number of high level conceptual templates (D'Antonio et al., 2007).

Remaining Challenges

Despite much progress, many challenges remain to helping scientific researchers develop and make better use of ontologies. Templates, patterns, and tools can support the ontology engineering process. However, communities such as Biomedicine and GeoSciences have somewhat different histories and tackle different problems. As a result, best practices are not yet employed for some domains, with research teams in these areas having a learning curve to realize best practices.

It remains to be seen, for example, if we can reuse templates from different domains such as biomedicine, industry, finance and geoscience. In OBO a particular, specialized group may set up its specific template for use and similar things may have to be done in other domains. In all domains there are some best practices that only experienced ontologies are likely to handle such as how to reify a time sensitive relation or apply the idea of a role relation.

To some degree the US effort on ontologies has been concentrated in the biomedical realm while in the EU the effort seems more concentrated on training a new generation on techniques that may then be applied more widely (Hitzler, 2022). The OBO community is engaged with large ontologies while domains outside of this may be looking at schemas with a modest number (thousands) of entities and in some cases employing lightweight axioms. Another issue is the scope of a templated design pattern. Reusable components need to be large enough to not be obvious, yet small enough to not be overwhelming and difficult to understand. Currently OBO templates are relatively small, while something like SOSA is much larger but still manageable. It is reasonable to expect that more domains will soon be faced with scaling problems, just as the OBO community has.

Current ontology engineering methods are too low-level. Regardless of the size of an ontology, we need higher-level approaches as part of the simplification. Currently basic semantic web tools (OWL-API and reasoners that are part of Protégé) were used to create the large KWG. However, reasoners may be overrated for some tasks; the Shapes Constraint Language (SHACL), a W3C recommendation for describing and validating RDF graphs, may be more effective than OWL. SHACL can be used to constrain elements of a KG by specifying that all instances of a particular class must have a certain property, or that the values of a particular property must be drawn from a specific list.

For ontology editing mid-level languages such as Turtle or the Manchester syntax, make the textual inspection and editing of ontologies easier, but graphical visualization remains an important enabler in many practical tasks related to ontologies especially as ontologies have grown in size and complexity. The Protégé editing tool includes a basic visualization capability (Protégé visualization, 2023) and plugins, such as OntoGraph

or OWLViz. Ontology libraries such as BioPortal also provide visualization as do the plugins that are part of the NeOn Toolkit. Tools such as VOWL allows some degree of useful structural inspection, but not visual editing or readily showing the structural consequences of changes. Standalone tools can be downloaded and installed on an end-user system, while web-based visualization tools enable the user to upload or select the ontology they want to visualize. Some examples of other ontology visualization tools are OWLGrEd, NavigOWL, Knooks, and TGViz (Joseph and Lourdusamy, 2020). Some tools allow a particular graphical representation such as UML-based diagrams into OWL. Despite the ability of plugins to allow new functionality there remains a need for some general ontology visualization framework that guides how to move from a conceptual model to ontology coding (Dudáš et al., 2018). Such a framework would include a core set of visual and interactive features that would be harmonized with templates. As with the system of OTTR templates that would allow extension and customization from a base to serve a family of related use cases. Better conceptual visualization of knowledge is also needed to aid in reaching domain agreements and arriving at working knowledge.

A large challenge is to establish and maintain harmonization across a range of semantic resources. Definitions are an essential part of ontologies, but early in an ontology's development they are often poorly written, incomplete or rigid and thus hard to formalize in a useful way. Some ontology efforts find reuse difficult and start anew using their own ideas. In many domains the practice of reusing extant ontologies for particular applications remains difficult, since the scope of the original ontologies competency questions may not have good overlap with a new application. Under these circumstances it is often easier to build new ontologies. Addressing this problem remains an issue since the result is various knowledge silos across the semantic spectrum that include definitional silos in glossaries which involve alternative conceptualizations. As a result, ontologies may suffer because of a lack of experience with writing definitions or simply because there is a lack of emphasis on properly defining terms during the ontology development process. Semantic resources for definitions sources, such as glossaries, can help but may vary widely on the same topic and are often inconsistent with one another. The challenge of agreeing on common conceptualizations across a domain and the wide range of semantic

resources that must be harmonized has been extensively been discussed in the Ontology Summit 2021 Communiqué (Baclawski et al., 2022).

Vocabulary alignment is challenging because:

- Arduous review is necessary due to the need to consider every term.
- Granularity of terms frequently differ (broader, narrower, ambiguous)
- Definitions/descriptions usually differ. For example, terms may be related but reconciling differences can be challenging.
- Formal axioms are often different.

For all these reasons, harmonization of concepts is important throughout the ontology lifecycle as an ontology is updated and made interoperable with other ontologies (Baclawski et al., 2022). And there remains a challenge to develop better tools for harmonization.

Discussion of Tool Issues

There are many and varied tools at different levels of maturity to consider, making it difficult to know how to judge their value. Only a few evaluations of ontology tools have been conducted (Duineveld et al., 2000; Malik, 2017). These efforts are somewhat dated and do not systematically cover the full range of evaluation from:

- Functionality across the lifecycle,
- Performance such as execution time,
- Usability such as interface robustness,
- Ease of learning,
- Intuitiveness, overall user experience
- Interoperability such as integration with other software tools commonly used in the ontology engineering process, and
- Community adoption.

However, in the BioMedical realm there are some tool overviews. These include Python tools (Semantic Python Overview, n.d.). a curated list of ontology resources including tools (Awesome Ontology, n.d.), and a

range of tools across different development packages (Awesome Semantic Web, n.d.).

Within the engineering lifecycle adequate tool development and maintenance is still needed over time. This reflects in part a prototype problem as argued by Vigo, Matentzoglu, Jay and Stevens (2019). Too many ontology authoring tools remain prototypes that are not widely used by groups beyond a narrow domain scope. Tool maturation requires funding such as was the case for Protégé tooling.

There is also an issue with how well integrated tools might be to an engineering methodology such as modular development.

ML/AI Techniques

Machine learning techniques have advanced rapidly and knowledge sources from text extraction, such as vector space embeddings, are now often used in developing KGs to represent the same type of artifacts as ontologies: entities, relationships, and attributes. Such embeddings capture semantic relationships and similarities between entities, enabling various graph-related tasks independently of what is formalized in ontologies. The emerging use of Knowledge Graph Embeddings as input features of machine learning methods has given even more visibility to this kind of representation, but raises new issues of understandability and interpretability of such embeddings. These include what the embeddings represent, understanding the logical connections between the graph and its embeddings and how they relate to the structure and semantics of what people understand and have formalized in the semantic resources that may underlie the KG.

An active area of research focuses on how to integrate traditional symbolic representations,as found in ontologies, with machine learning approaches. There is a rich possibility of unifying these 2 approaches (symbolic and connectionist) , symbolic and connectionist approaches, with different representations as well as reasoning. There are ongoing efforts to study the compatibility between ontological knowledge and different types of vector space embeddings (Kulmanov et al., 2020).

Ontology Maturity

Maturity of Ontology has been proposed by several authors (Obrst, 2009; Doerr, 2014). While Obrst refers to linkages with databases as level

3 and axiomatized, inference, persistent management of domain semantics at level 5, Doerr refers to A. Scope, B. Domain based answers to research questions with constant terms, C. Classes as carriers of properties.

Open extensible ontologies need to be measured by not only semantic richness but also other measures such as usability by domain experts (scientists and practitioners), related domains-overlap and cross-domain relevance (may be indices!), and reuse by applications.

While the Zachman framework has been extensively used by Industry, Federal and State Entities including Defense for describing the knowledge of an enterprise, it can also be described as akin to an irreducible description capturing the state of Enterprise (Zachman framework, n.d.; Sharma, 2000). The use of the Zachman framework includes many areas of applications, standards, tools, technologies, vocabularies and objectives, and success related to maturity is in actual implementation of enterprise. While very mature data captures about the enterprise are kept in master-data and metadata, this interpretation of the Zachman framework implies maturity of the enterprise in terms of extent of knowledge and progress in terms of AS-IS and TO-BE gap analysis, and how to reach the target state. We are attempting a similar gap analysis notion in this communiqué for ontology tools and their domain use and developing notions of maturity as we do for software engineering.

Ontologies may also be used for Enterprises such as Finance (e.g., the Financial Industry Business Ontology) and Manufacturing (e.g., the Zachman framework). An example of maturity in the Zachman framework is measured by the levels reached across rows and columns and most matured Enterprises include implemented models including items such as BPM, Standards adherence and Reuse through replication of tools, standards and infrastructure. The maturity in the development phases of ontologies can be compared to Capability Maturity Model Integration (CMMI) maturity levels (CMMI, n.d.). The final test of ontology-maturity is perhaps related to Logic, Reasoning, Inference and Cognitive power and visualization through tools such as Knowledge Graphs, Formal Notations such as BPMN, other metamodels, etc. There are other methods such as Gartner Hype Cycles and Forreser studies that represent the maturity and life cycles of tools, technologies and standards, eventually predicting

the operational and sunset phases. We expect these tools to have different peaking and operational phases and eventual replacement by better solutions!.

Summary and Conclusion

We have shown how one can mitigate the difficulties that typical end-user researchers encounter when developing and employing ontologies. These include techniques for initiating the development of an ontology, examples of tools tools that can be used, and how to maintain an ontology after its development. Higher-level work on ontologies can begin by securing cross community cooperation, agreements and common tool use. The ontology reuse, templates and design patterns are important techniques. While significant progress has been made, these it is still challenging to make effective use of these techniques Addressing this challenge requires a broad, yet in-depth, analysis of ontologies, ontology design patterns, and their potential to provide a usable, common basis that others can effectively use.

Templated abstractions can be used with formalisms such as RDF and OWL. Doing so allows non-ontologists to use, capture, and instantiate standard abstracted patterns for activities such as type checking. Standardized formats need to be adapted for domain experts, data managers and ontology experts. The templates and ODPs can be published in template libraries using LOD principles and related tools that must be open source and well managed. However, it remains challenging to design patterns that are general enough to be able to cover a wide variety of applications and uses. Of course, it is necessary to bear in mind that there can be cultural, operational, ideological and political differences among the communities that employ ontologies, as well as varying domain challenges. Consequently, continued work on the adaptation of templates for new ontology development will be needed.

Acknowledgments

We wish to acknowledge the contributions to this communiqué by Veda C. Storey, Nomi Harris, Robert Rovetto, Cogan Shimizu, Ravi Sharma, Asiyeh Yu Lin, Ken Baclawski, Ram Sriram, and Alex Shkotin. We also appreciate the efforts by the speakers and participants in the summit sessions.

References

Alani, Harith, et al. (2002). "Managing reference: Ensuring referential integrity of ontologies for the semantic web." Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web: 13th International Conference, EKAW 2002 Sigüenza, Spain, October 1–4, 2002 Proceedings 13. Springer Berlin Heidelberg.

Awesome Ontology download site (n.d.). https://bit.ly/3G6xqYt

Awesome Semantic Web download site: Python (n.d.) https://bit.ly/3slndE2

Baclawski, Ken, et al. (2021) "Ontology Summit 2020 Communiqué: Knowledge Graphs." *Applied Ontology* 16(2): 229–247. https://bit.ly/34fFVOi

Baclawski, Ken, Michael Bennett, Gary Berg-Cross, Leia Dickerson, Todd Schneider, Selja Seppälä, Ravi Sharma, Ram D. Sriram, and Andrea Westerinen (2022). "Ontology Summit 2021 Communiqué: Ontology Generation and Harmonization." *Applied Ontology* 17(2): 233–248. https://bit.ly/3mhNlIP

Bennett, Mike, and Baclawski, K. (2017). "The role of ontologies in linked data, big data and semantic web applications." *Applied Ontology* 12(3-4): 189–194.

Berg-Cross, G., Duerr, R. et al. (2023). "Semantics Adoption in the Earth and Environmental Sciences: Successes and Roadblocks" https://bit.ly/3G0trg1

Bioportal website (n.d.). https://bit.ly/3u4edUx

Bioregistry website (n.d.). https://bioregistry.io

CMMI (n.d.). https://cmmiinstitute.com/

Carriero, V. A., Gangemi, A., Nuzzolese, A. G., and Presutti, V. (2019). An Ontology Design Pattern for representing Recurrent Events. In Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, K. Janowicz, AA Krisnadhi, MP Villalón, K. Hammar and C. Shimizu, eds, CEUR Workshop Proceedings, volume 2459. https://bit.ly/45Zkm1j

D'Antonio, Fulvio, Michele Missikoff, and Francesco Taglino (2007). "Formalizing the OPAL eBusiness ontology design patterns with OWL." Enterprise Interoperabil-

ity II: New Challenges and Approaches. Springer London.

Doerr, M. (2014). Answer to "Is there a maturity model for ontologies?" Research Gate. https://bit.ly/40tDeUU

DOSOP, 2023. https://bit.ly/49hwWMa

Dudáš, Marek, et al. (2018). "Ontology visualization methods and tools: a survey of the state of the art." *The Knowledge Engineering Review* 33: e10.

Duineveld, Arthur J., et al. (2000). "WonderTools? A comparative study of ontological engineering tools." *International Journal of Human-Computer Studies* 52(6): 1111-1133.

FAIR Principles (n.d.). https://www.go-fair.org/fair-principles/

Fürber, Christian, and Christian Fürber. (2016). Semantic technologies. Springer Fachmedien Wiesbaden.

Grüninger, Michael, Uschold, Michael and Guarino, Nicola, Application Framework Synthesis (2011), in Uschold, Michael. "Making the case for ontology." *Applied Ontology* 6(4): 377-385.

Hahmann, Torsten, Shirly Stephen, and Boyan Brodaric (2016). "Semantically refining the groundwater markup language (GWML2) with the help of a reference ontology." International Conference on GIScience Short Paper Proceedings. 1(1).

Hahmann, Torsten (2023). Upper Level Domain Models, Briefing at the Ontology Summit 2023. http://bit.ly/3KgnvCg

Hitzler, Pascal (2022). Panel on FAIR semantic resources, organized by Gary Berg-Cross and Asiyah Lin, 4th U.S. Semantic Technologies Symposium, October, 2022.

Hitzler, Pascal, and Cogan Shimizu (2018). "Modular ontologies as a bridge between human conceptualization and data." Graph-Based Representation and Reasoning: 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings 23. Springer International Publishing.

Hoyt, C.T., Balk, M., Callahan, T.J. et al (2022). Unifying the identification of biomedical entities with the Bioregistry. *Sci Data* 9: 714. https://bit.ly/3sc1QFm

Hoyt, C.T., Hoyt, A.L. and Gyori, B.M. (2023). "Prediction and curation of missing biomedical identifier mappings with Biomappings." *Bioinformatics* 39.4: btad130.

Jackson, Rebecca C., et al. (2019). "ROBOT: a tool for automating ontology workflows." *BMC bioinformatics* 20: 1-10.

Jackson, R., Matentzoglu, N., Overton, J. A., Vita, R., Balhoff, J. P., Buttigieg, P. L., ... and Peters, B. (2021). OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database*. https://bit.ly/47075XK

Janowicz, Krzysztof, et al. (2022). "Know, Know Where, KnowWhereGraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence." *AI Magazine* 43(1): 30-39.

Joseph, M. F., and Lourdusamy, R. (2020). Feature analysis of ontology visualization methods and tools. *Computer Science and Information Technologies*, 1(2), 61-77.

Katifori, Akrivi, et al. (2007). "Ontology visualization methods—a survey." *ACM Computing Surveys* (CSUR) 39(4): 10-es.

Krieg-Brückner, Bernd, Till Mossakowski, and Mihai Codescu (2021). "Generic Ontology Design Patterns: Roles and Change Over Time." WOP (Book).

Kulmanov, Maxat, et al. (2020). "Machine learning with biomedical ontologies." biorxiv: 2020-05.

Linked Open Vocabularies (n.d.). https://bit.ly/3Qoijy9

LinkML website (n.d.). https://linkml.io

Malik, Zeeshan Haider (2017). "Usability evaluation of ontology engineering tools." 2017 Computing Conference. IEEE.

Matentzoglu, Nicolas, et al. (2022). "Ontology Development Kit: a toolkit for building, maintaining and standardizing biomedical ontologies." *Database*: baac087.

Musen, Mark A., et al. (2018). "CEDAR: semantic web technology to support open science." Companion Proceedings of the The Web Conference 2018.

Neuhaus, F. and Smith, B. (2010). Ontology Summit 2010 Communiqué: Creating the Ontologists of the Future. http://bit.ly/2fOGuIO

Neuhaus, F. et al. (2011) Creating the ontologists of the future, *Applied Ontology* 6(1) 91–98. https://bit.ly/3FOU3R2

Noy, Natalya, et al. (2010). "The ontology life cycle: Integrated tools for editing, publishing, peer review, and evolution of ontologies." AMIA Annual Symposium Proceedings. Vol. 2010. American Medical Informatics Association.

OBO (2023). Open Biological and Biomedical Ontology Foundry, Community development of interoperable ontologies for the biological sciences. http://obofoundry.org/

Obrst, L. (2009) The Ontology Maturity Model. https://bit.ly/3CUAqDs

Obrst, L. and Musen, M. (2008) Ontology Summit 2008 Communiqué: Towards an Open Ontology Repository. http://bit.ly/2fOGs3E

OQUAT (n.d.), Ontology Quality Assessment Toolkit. https://bit.ly/471g2QD

Otte, J. Neil, John Beverley, and Alan Ruttenberg (2022). "BFO: Basic formal ontology." *Applied Ontology* 17(1): 17-43.

Protégé visualization (2023). Visualization - Protégé Wiki (stanford.edu)

Rudnicki, R., Smith, B., Malyuta, T., and Mandrick, C. W. (2016). Best Practices of Ontology Development, October 25, 2016. CUBRC: Buffalo, NY, USA.

Semantic Python Overview download site (n.d.). https://bit.ly/40tCX4k

Shimizu, Cogan, and Karl Hammar (2019). "Comodide–the comprehensive modular ontology engineering IDE." ISWC 2019 Satellite Tracks (Posters and Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019) Auckland, New Zealand, October 26-30, 2019.. Vol. 2456. CEUR-WS.

Shimizu,C., Hitzler, P and Krisnadhi, A. (2020). "Modular ontology modeling: A tutorial." In Cota, G., Applications and Practices in Ontology Design, Extraction, and Reasoning, *Studies on the Semantic Web*. Volume. 49.

Shimizu, Cogan, Karl Hammar, and Pascal Hitzler (2023). "Modular ontology modeling." *Semantic Web*, 14(3): 459-489.

Shimizu , Cogan (2023). Introduction to Patterns and Modules, Ontology Summit 2023.

Skjæveland, Martin G., et al. (2018). "Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates." The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I 17. Springer International Publishing.

Skjæveland, Martin G. (2023). "Reasonable Ontology Templates (OTTR)," Ontology Summit 2023 presentation, April 19, 2023. https://bit.ly/3KHDmbW

Sorbonne Declaration (2020). Sorbonne Declaration on Research Data Rights. https://sorbonnedatadeclaration.eu/

Suárez-Figueroa, Mari Carmen, Asunción Gómez-Pérez, and Mariano Fernandez-Lopez (2015). "The NeOn Methodology framework: A scenario-based methodology for ontology development." *Applied Ontology* 10(2): 107-145.

Zachman framework (n.d.). The Zachman framework. Wikipedia. https://bit.ly/4756zI6

Appendix 1: NeOn glossary of terms

Ontology network A set of ontologies that are related to each other and that are used to represent a domain of interest

Ontology A formal, explicit specification of a shared conceptualization

Ontology engineering The process of creating, maintaining, and using ontologies

Reuse The use of existing ontologies or parts of ontologies to create new ontologies

Reengineering The process of adapting existing ontologies to meet new requirements

Collaboration The working together of people to achieve a common goal

Distributed environment An environment in which people are working together on a project but are not located in the same place

Scenario A description of a typical use case for an ontology network

Appendix 2: Phases of work and NeOn Scenarios

In NeOn seven phases of work are used, as shown in Figure 3, starting with an Initiation Phase that creates a requirements Specification and project Scheduling. As in true of all phase evaluation of the scheduled work, early phase follows a rational project path and defines the problem by:

1. Identifying the criteria used to judge possible solutions (e.g., functional requirements and/or competency questions for ontologies or alignment with a certain vocabulary to provide interoperability)
2. Deciding how important each criterion is (e.g., Must this ontology use the Common Ontology for BioMedicine as a starter set? Or "For efficiency should I use a ROBOT template?")
3. Generating a list of possible alternatives
4. Evaluating the possible alternatives before making a selection.

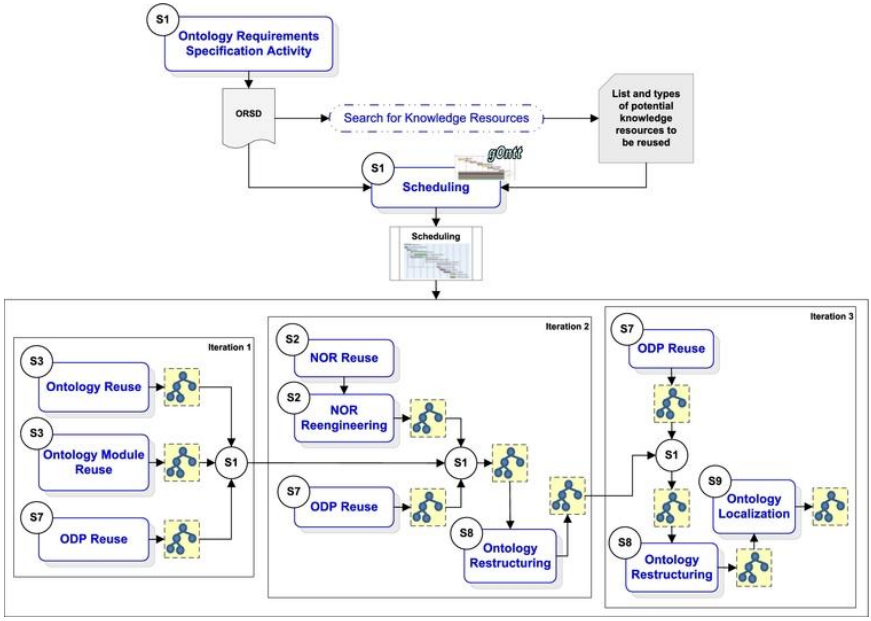

Figure 3: Graphical depiction of several different steps in ontology development, where each step has its methods and interactions with other steps (Suárez-Figueroa et al., 2015)

Appendix 3: The Reasonable Ontology Templates and Methodology

The OTTR templates are accessible through a template library hosted at https://tpl.ottr.xyz/, following linked data principles. These templates offer a more intuitive and abstract approach to ontology definition by employing a range of related templates for conceptualizing entities. In addition to compositional patterns akin to those used in OBO, OTTR provides templates for triples and class typing, serving as alternatives to the direct composition of complex OWL code. Furthermore, OTTR templates address OWL language-specific challenges such as handling blank nodes and default values.

Each OTTR template begins with a unique identifier and a descriptive name, facilitating top-down modeling that starts with a high-level, conceptual perspective of the ontology's structure. This initial step involves

identifying and defining the pivotal concepts and relationships to be captured within the ontology. Since not all concepts are initially known, as the process evolves, templates can be utilized iteratively. Domain experts can contribute to a spreadsheet containing their proposed concepts, which is subsequently expanded to encompass relationships, properties, and other pertinent elements.

By leveraging OTTR templates, ontology developers can streamline their modeling workflow, enhance collaboration with domain experts, and progressively refine their ontologies by iterating through concept expansion and relationship establishment.

Following base development users arrive at the more demanding task of class restrictions. These are progressively refining and expanding it through creation of high-level OTTR templates that capture the structure and constraints of the ontology components. Templates in OTTR also support parameterization, which allows users to define patterns with placeholders that capture different aspects of ontology elements and provide flexibility in generating OWL code for repetitive ideas. Templates that can be populated with specifics although there may be issues such as global fields which are often inadequate for common items. For example one gets name collisions for ambiguous,common words such as “member” or “time” that might be used in populating a template. Domain terms such as “equipment” need specific restrictions like “equipment can only have equipment as a part.”

OTTR templating also enables a clear separation of ontological representation concerns by distinguishing template definitions that capture the structure and constraints of ontology components, from the template instantiations which allows users to populate the templates with relevant information, such as class names, property values, and individual instances. This separation simplifies the process of modifying or extending ontologies without directly modifying the base code.

Following initial instantiation, users can iteratively refine the ontology by expanding and modifying the templates based on the evolving requirements and domain understanding. This refinement process involves adding more specific details, defining additional relationships, and incorporating domain-specific constraints.Users can instantiate templates again with updated data to generate expanded code. This repeated instantiation and ex-

pansion process allows users to incorporate the changes and improvements made during the iterative refinement step. Finally to ensure the correctness and integrity of the ontology validation and testing is supported by testing the evolving ontology code against domain constraints, consistency rules, and desired ontological principles.

Importantly OTTR templates are designed to handle the underlying semantics of OWL, which means they can enforce certain constraints or rules during code generation. This semantic-awareness helps in maintaining the integrity and consistency of the resulting OWL ontologies. The code generation process built around OTTR templates resolves parameters and applies the provided data to the templates, to produce a final, usable OWL representation.

Templates can also serve as a simple checklist to ensure that important, core aspects of the ontology have been considered during design.

GARY BERG-CROSS is a cognitive psychologist (PhD, SUNY-Stony Brook) whose professional life included teaching and R&D in applied data and knowledge engineering, collaboration, and AI research. A board member of the Ontolog Forum he co-chaired the Research Data Alliance workgroup on Data Foundations and Terminology. Major thrusts of his work include reusable knowledge, vocabularies, and semantic interoperability achieved through semantic analysis, formalization, capture in knowledge tools, and access through repositories.

JAMES A. OVERTON is an open source software and ontology developer, specializing in data integration for open science databases. As a member of the OBO Technical Working group he builds and maintains tools and infrastructure for an open community of more than 200 scientific ontology projects. As president of Knocean Inc. he and his team have helped scientific databases at NIEHS, NIAID, EBI, universities, and other institutions use ontologies to integrate, validate, analyze, and share FAIR data.