Ontology-Driven Software: What We Learned From Using Ontologies As Infrastructure For Software *Or How Does It Taste to Eat Our Own Dogfood*

Csongor Nyulas, Natalya F. Noy, Michael Dorf, Nicholas Griffith, Mark A. Musen

Stanford University, Stanford, CA 94305, US
{nyulas,noy,mdorf,ngriff,musen}@stanford.edu

Abstract. In recent years, researchers have argued that using ontologies to represent and drive knowledge infrastructure of software projects provides separation of the declarative and procedural knowledge and enables easier evolution of the declarative knowledge. We have validated these conjectures in the context of BioPortal, a repository of biomedical ontologies, which was developed in our group. We are using the BioPortal Metadata Ontology to represent details about all the ontologies in the repository, including internal system information and the information that we collect from the community such as mappings between classes in different ontologies, ontology reviews, and so on. To the best of our knowledge, BioPortal is the first large-scale application that uses ontologies to represent essentially all of its internal infrastructure.
The BioPortal Metadata Ontology extends several other ontologies for representing metadata, such as the Ontology Metadata Vocabulary and the Protégé Changes and Annotations Ontology. In this paper, we show that it is feasible to describe the structure of the data that drives an application using ontologies rather than database schemas, which are used traditionally to store the infrastructure data. We also show that such approach provides critical advantages in terms of flexibility and adaptability of the tool itself. We demonstrate the extensibility of the approach by enabling representation of views on ontologies and their corresponding metadata in the same framework.

1 Representing Knowledge Infrastructure: From Database Schemas to Ontologies

The topic of using Semantic Web technology to facilitate software development, integration, and evolution has been an active area of Semantic Web research, with annual workshops on Semantic-Web Enabled Software Engineering (SWESE). Researchers have pursued several different directions in this line of work: generating software code from ontologies [7] or using ontologies to describe inputs, outputs, or tasks of software components to enable integration of software and services [12]; facilitating gathering of requirements from domain experts [1]; using ontology-based reasoning to validate integrity and consistency of software models [2]; or facilitating critical software-engineering tasks, such as configuration management [16] and product management [8]. These novel applications, as well as the traditional ones, use ontologies to describe only some of the artifacts, whereas the structure of the rest of the data is reflected in a

database schema. In this paper, we describe an approach to application development that pushes this envelope to use ontologies and ontology instances to represent essentially *all* data that the application requires in a single flexible framework, from declarative high-level descriptions of the data as in the examples above to internal system data.

For large-scale distributed architectures today, the development stack includes several technologies wrapped around a SQL database schema, such as persistence managers (e.g., Hibernate), a web server, and so on. When the database schema changes, these changes often need to be propagated through the development stack, thus making such changes expensive in distributed web-based applications. We have encountered this problem in developing BioPortal[1]—a community-based repository of biomedical ontologies, containing 170 ontologies with more than one million classes among them at the time of this writing. Users can submit their ontologies to BioPortal; search across all ontologies; browse the ontologies, their different versions, and the associated descriptions and provenance information; describe their ontology-related projects and link the descriptions to the ontologies; leave comments on classes and on ontologies; create mappings between concepts in one ontology and concepts in another ontology [11].

The BioPortal application is heavily *knowledge-driven*: most of what the users see when browsing BioPortal (in addition to the ontologies themselves), is some rendering of information that would traditionally be in a database. This internal information that drives the application includes the metadata about ontologies in the repository, such as ontology domain, authors, and other provenance information, as well as information on which property to use for preferred name and synonyms in each ontology, information on where the ontology itself resides in the system (e.g., the specific database table), when it was uploaded, the name of the administrator of the ontology in BioPortal, and so on. Some of this information (such as provenance) is intrinsic to the ontology artifact and is relevant outside of BioPortal; some information is internal system information.

Because the BioPortal application is novel in many of its aspects, our internal infrastructure continues to evolve constantly, as we understand better user requirements, learn what works and what does not, get new collaborators that would like to extend BioPortal in a certain way. With the knowledge infrastructure constantly in flux, we found that describing and representing the structure of the knowledge as a relational database schema did not provide the flexibility and quick adaptability that our users required. Making changes was cumbersome and put a bottleneck in the development of the software code. It also made it much harder for anyone to adapt the BioPortal code for their own purposes as the developers had to be familiar with the entire development stack (including Protégé, Java, Spring, Hibernate, and Ruby-on-Rails).

Thus, we decided to "eat our own dog food:" we developed an ontology to describe this infrastructure and represented the application data itself as ontology instances. Thus the whole BioPortal application is driven by ontologies and ontology instances. Note that while BioPortal is a repository of ontologies, the infrastructure that we describe is not specific to the artifacts represented in a repository. It will work for a repository of any other artifacts, not necessarily ontologies. To the best of our knowledge, BioPortal is the first large-scale application of this approach.

This paper makes the following contributions:

[1] `http://bioportal.bioontology.org`

– We developed an ontology to represent the infrastructure and run-time data of a large community-based ontology repository.
– We implemented the infrastructure of BioPortal using an ontology to represent most of the data required to drive the application.[2]
– We validated the extensibility of the approach by adding functionality to support flexibly representation of ontology views.

2 Types of Metadata in the Repository

The BioPortal ontology repository is an active ontology repository with a large user community that contributes its content and uses its web services in their applications. In addition to more than 170 ontologies, it currently contains multiple versions of these ontologies, submitted by their authors and almost one million mappings between concepts in the ontologies. There are descriptions of ontology-based projects, and notes and discussions on classes and ontologies. The BioPortal Resource index provides ontology-based access to several biomedical data sets available online (e.g. entries in GEO, ClinicalTrials.gov). All BioPortal functionality is supported by a rich metadata infrastructure, which includes the following types of metadata:

– **ontology metadata** describing the ontologies and their provenance and includes ontology name, domain, description and keywords, authors, license information, versions, references, and metrics such as the number of classes and properties;
– **mappings** between concepts, and metadata associated with mappings, such as how the mapping was created, whether it was created manually or computed automatically by a particular algorithm (and which one) and context for the mapping [10];
– **ontology reviews** are contributed by users as part of their evaluation of ontologies in BioPortal;
– **notes on classes** are user-contributed notes that can contain questions, comments, and suggestions, usually addressed to the authors of specific ontology classes;
– **projects** that use ontologies, described by BioPortal users;
– **user information** such as user profiles, information on who administers each ontology and each project description, who contributed notes, mappings, and reviews to BioPortal, and so on.

3 Functional and Architectural Requirements for Metadata Support

The BioPortal application dictates the following functional and architectural requirements for the metadata support:

Efficient and scalable support of BioPortal main functions: Any metadata infrastructure must support fast access to metadata, flexible querying of specific metadata items and their combination, and be scalable. We envision that the number of users, notes, projects, and mappings will grow significantly in the coming months.

[2] At the time of this writing, some data, such as mappings and user information, is still in database tables from our initial implementation of BioPortal.

Support for ontology versioning: Users can upload successive versions of their ontologies and explore any ontology version. There must be services that always resolve to the latest version of an ontology, with each ontology having a "virtual" location that always redirects to the latest version. Metadata referring to an ontology or its components (e.g., reviews, notes, mappings) must be attached to a specific version of an ontology.

Flexible evolution of the metadata schema: One of the key requirements for metadata support is its ability to adapt easily to new requirements and types of metadata. The types of metadata that an ontology repository requires is still an active area of research. Thus, the structure of the metadata and the specific properties change frequently. These changes to the schema describing the metadata must be easy to implement and roll out.

Customizability of the metadata schema: Some groups install their own versions of BioPortal software to support either a broader scope than just biomedicine (e.g., the Open Ontology Repository sandbox[3]) or to maintain a repository open only to a specific set of users (e.g., the Marine Metadata Initiative[4]). Developers that maintain these BioPortal installations usually customize the code to satisfy the local requirements. For example, the fields that describe an ontology and its provenance are different for different communities. Definitions of mappings and the associated metadata differ as well. The representation of metadata schema must make it easy for these developers to custom tailor what gets represented and what gets presented in the user interface.

Reuse of existing technologies and ontologies: Wherever possible, we would like to use existing technologies and standards for representing metadata. For example, the Ontology Metadata Vocabulary (OMV) [14] provides a vocabulary for describing ontologies. There are several ontologies and APIs for describing mappings (e.g., the alignment API [4] or the Protégé mapping ontology [10]). Reusing these ontologies enables us not only to use technologies that have already been tested but also to share the data represented using these ontologies. For example, by using OMV to represent ontology metadata, we can share these descriptions with other repositories that use OMV, such as Oyster [13] and Cupboard [3]. Similarly, the comments on ontologies that BioPortal users provide are useful for ontology authors when they evolve their ontology. In order for ontology authors to see these comments alongside the ontology classes in their favorite ontology-editing environment (such as Protégé). Thus, the comments must be represented in the format that an ontology editor, such as Protégé, can understand (e.g., the Protégé Changes and Annotations Ontology, CHAO [9]).

In our initial implementation of BioPortal we used a database schema to describe our metadata, with column names corresponding to metadata fields. This approach is fairly traditional for many large-scale implementations and supports the first two requirements in our list—efficient and scalable handling of metadata and support for ontology versioning. However, in our experience, this approach did not fare so well on other requirements.

Any time we needed to add a new metadata field, we had to change not only the database schema, but also the rest of the application stack (e.g., Hibernate) to reflect the

[3] `http://oor-01.cim3.net/home/release`
[4] `http://mmisw.org/or/`

change. These changes were time-consuming and cumbersome. Yet, the more we were developing BioPortal, the more features we were adding, the more often we needed to adjust the metadata representation. For instance, in addition to describing the ontologies themselves, we needed to add ontology views, to support ontology reviews along several different dimensions, and to represent a large set of ontology metrics—all of these requirements crystalized after the start of the development.

With the metadata schema encoded as a database schema, any customization of new BioPortal installation requires changes to the schema as well. And, as we mentioned earlier, this process is cumbersome and error-prone.

Reusing and sharing the metadata that we collect also was not straightforward: it is hard to find two applications that use the same database schema. Thus, in order to transform the metadata from our internal representation to the representation that another repository (e.g., Oyster) uses or to represent comments in a format that a tool such as Protégé would understand, we must write a script to export the data.

While none of these challenges are insurmountable, we decided to apply a completely different approach to representing metadata infrastructure in BioPortal to address these requirements. As we show in the remaining sections, this approach satisfied our requirements and proved to be extensible enough to support new requirements.

4 Architecture

Figure 1 shows the architecture of BioPortal. It is a traditional service-oriented layered architecture, with the front end (Ruby-on-Rails) accessing the backend information through RESTful services.[5] There are services to access ontology information (e.g., get information about a specific ontology, upload a new version, get a diff between two versions), concept-level services (e.g., get class definition), hierarchy services (e.g., get all subclasses of a class), search services (e.g., search for a term across all ontologies), and other services. The business logic tier implements these services in Java, using the Spring framework. This layer is the one that contains the logic to translate the internal metadata representation into responses to service request (e.g., a service may request a list of all versions for a specific ontology). Before we transitioned to the ontology-based approach, the database schema of the underlying relational database (mySQL) was reflected directly in the implementation of this layer and its metadata functions. In our current approach, the metadata structure is accessed through the Protégé ontology API. This API, in turn, does use a database to store the ontology and the instances. However, the schema of the database that the Protégé uses does not depend on the ontology itself. It is a single table that stores both the ontology and the instance information.[6]

The types of metadata and their properties are describe in the **BioPortal Metadata Ontology**.[7] The metadata values are Protégé instances and property values (see Figure 2 for an example). We use the Protégé API to access the ontology and instances.

[5] `http://bioontology.org/wiki/index.php/BioPortal_REST_services`
[6] `http://protege.cim3.net/cgi-bin/wiki.pl?JdbcDatabaseBackend`
[7] `http://bioportal.bioontology.org/virtual/1148`

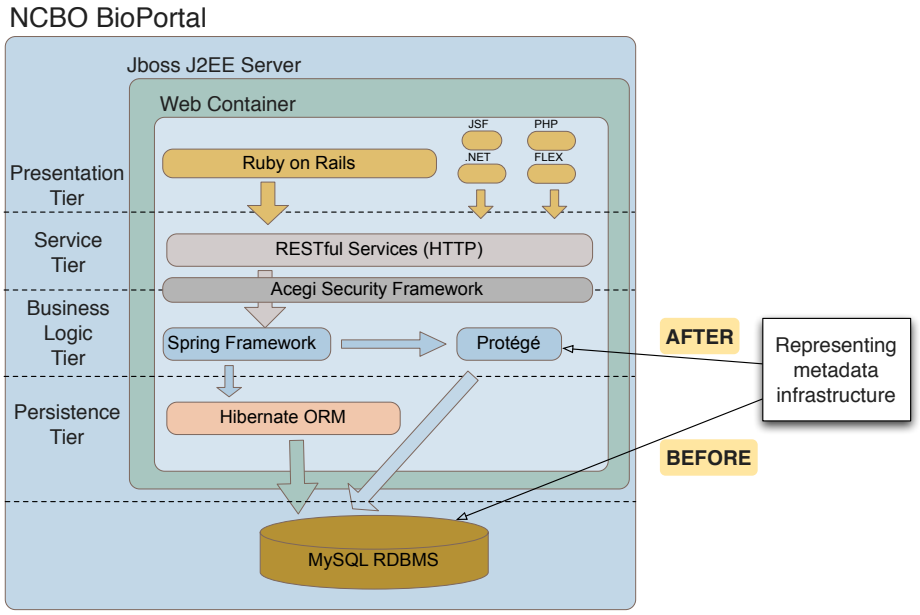

Fig. 1. Layered architecture of BioPortal: The Presentation Tier contains the user interface and external applications that call BioPortal REST services. The Business Logic Tier implements the logic of translating the data and metadata stored in the database into responses to the service calls.

5 The BioPortal Metadata Ontology

The BioPortal Metadata Ontology is an OWL ontology that imports a number of other ontologies (Figure 3) and includes classes to describe an ontology itself, its versions, information about the ontology, creators of an ontology, user-contributed content, such as notes, reviews, and mappings. It also contains the system information that is relevant for maintaining and representing the ontology in BioPortal, such as which users administer the ontology in BioPortal, where the ontology itself is located in the BioPortal system, internal ontology id and version ids, and so on. The instances of classes in this ontology represent the actual metadata for the BioPortal content. The BioPortal Metadata Ontologyis an OWL-Lite ontology, specifically, RDF Schema constructs, plus `owl:import`, thus supports in a scalable manner any reasoning that BioPortal requires (for example for transitivity in getting superclasses or subclasses of a class).

The BioPortal Metadata Ontology imports several ontologies that deal with the types of metadata that BioPortal supports:

- **The Ontology Metadata Vocabulary (OMV)** describes most of the metadata for ontologies themselves (e.g., domain, author, version, ontology language, etc.)
- **The Protégé Changes and Annotations Ontology** (CHAO) provides definitions for generic annotations and ontology components that they annotate.

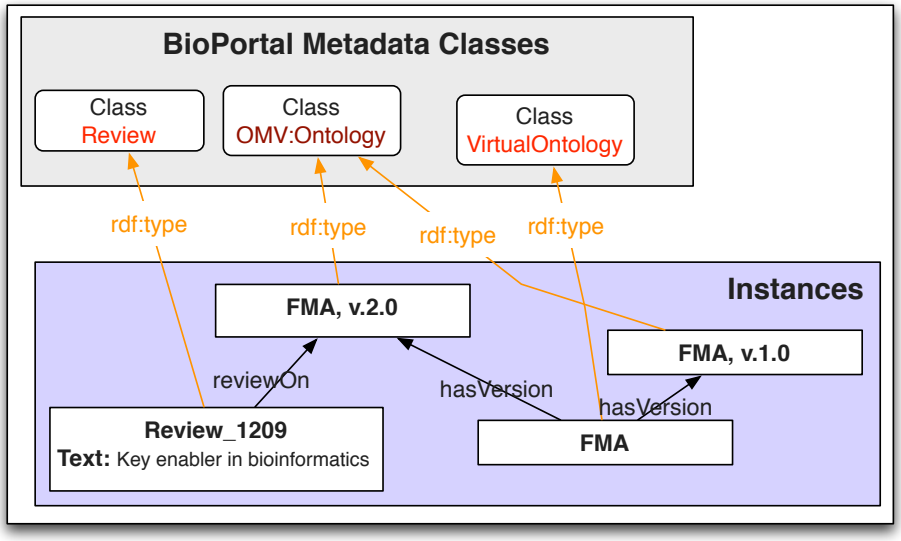

Fig. 2. Representing metadata as ontology and instances in BioPortal: The diagram shows some examples of classes and instances that represent metadata in BioPortal. There is an instance of the class `VirtualOntology` that corresponds to the Foundational Model of Anatomy (FMA) and not any specific version of it. This instance points to instances of `OMV:Ontology` describing specific versions of FMA in BioPortal. A review provided by a user (an instance of the class `Review`) points to a specific version of the FMA for which the review was created.

– **The Protégé Mapping Ontology** provides vocabulary for describing one-to-one mappings between concepts and corresponding metadata.

The OMV provides the vocabulary for describing a specific ontology version. An instance of the class `OMV:Ontology` describes a single version of an ontology. This class contains properties describing pertinent information about the ontology. The BioPortal Metadata Ontology extends this class to add properties that are specific to BioPortal as well as some missing properties that should have been in OMV.[8] These properties include system information such as the internal id, the user who submitted the ontology, the internal status (e.g., scheduled for parsing, loaded, error), and associated reviews.

We use the instances of the Protégé CHAO ontology to represent comments that BioPortal users contribute to the ontologies. Each comment is represented as an annotation attached to a specific class (in a specific ontology version) or to another annotation (if it is a response to a comment). Users can use the comments, for example, to carry out discussions about modeling decisions, make suggestions for changes, ask questions. The same mechanism exists in Collaborative Protégé, a version of the Protégé ontology editor that supports collaborative ontology editing. Because BioPortal and Protégé share the same structure for representing user comments and discussions, one can potentially

[8] We collaborate with OMV developers to include these properties in future versions of OMV

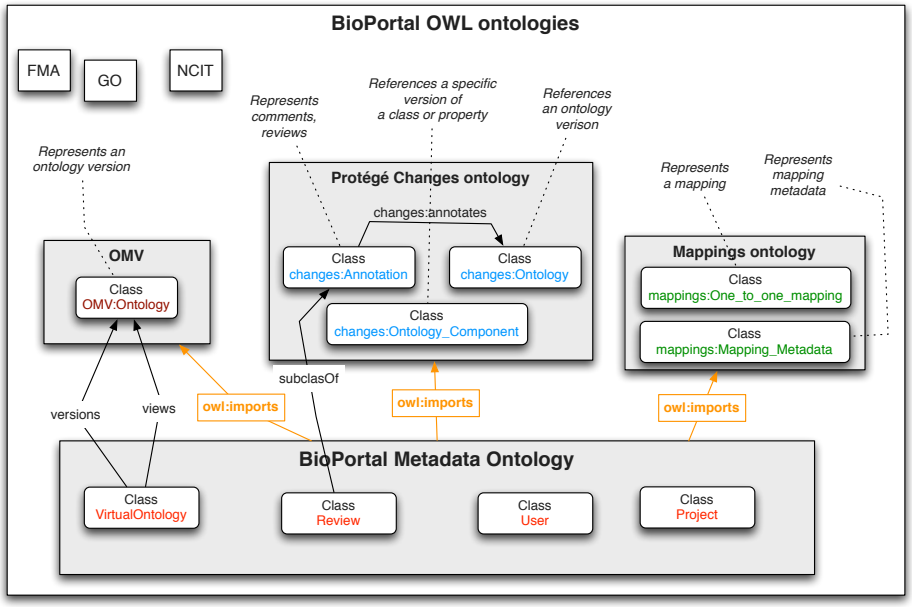

Fig. 3. The BioPortal Metadata Ontology: Some classes and ontologies that the BioPortal Metadata Ontology imports. The BioPortal Metadata Ontology is itself in the BioPortal repository, along with domain-specific ontologies such as the Gene Ontology (GO), the FMA, and others.

open a BioPortal ontology in Protégé and see the comments contributed by BioPortal users. We are currently working on the tighter integration of the two tools.

The BioPortal Metadata Ontology adds classes that are specific to BioPortal functionalities and that are not described in the imported ontologies. These classes represent ontology projects and reviews on ontologies and dimensions that users can use to evaluate ontologies.

The BioPortal Metadata Ontology introduces some convenience classes that abstract the information already present in other classes. This approach poses minor maintenance challenges, which we mitigate by changing instances only programmatically. However, this approach greatly facilitates access to the information. Consider the following example: In OMV, the description of an ontology version (instance of `OMV:Ontology`) points to a previous version of the same ontology. Thus, in order to present a table with the information about all versions of an ontology (as in Figure 4), we must first find the latest version of the ontology, and then traverse the instances to collect all versions. Thus, we introduce the notion of `VirtualOntology` which is an object representing the collection of all versions of the ontology. This object has the minimal information that is shared among all versions, such as the ontology name (e.g., Gene Ontology); the "virtual ontology id"—the global id that, when used to access an ontology, always resolves to the latest version; the information on who administers the ontology and the list of versions (instances of `OMV:Ontology`).

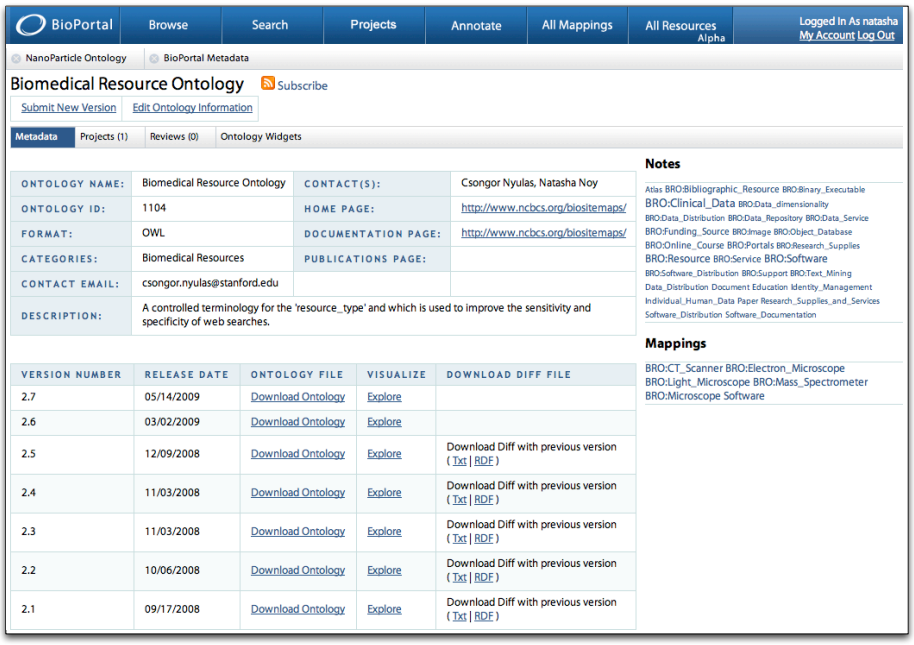

Fig. 4. Ontology metadata in BioPortal user interface: This page presents the metadata for one of the BioPortal ontologies. It shows the provenance and other information about the ontology itself, the list of different versions of this ontology in BioPortal, and links to notes that users contributed to this ontology and mappings between concepts in this ontology and concepts in other ontologies. The notes and mappings are presented as a tag cloud: classes that appear in a larger font have more notes and mappings than others.

6 Validating Feasibility: Implementing BioPortal metadata

We have validated the feasibility of our approach by implementing it as an infrastructure for BioPortal.[9] The previous version of BioPortal used a database schema to reflect the metadata schema and posed exactly the flexibility and customizability challenges that we described in Section 3.

First, we replaced the databases in the storage layer with a Protégé ontology, implemented in its own one-table ontology-independent schema (Figure 1). Second, we replaced the metadata implementation in the business-logic layer with the appropriate Protégé API calls. Third, we transferred the metadata from the old database to instances in the metadata ontology. Our goal was to maintain the same API at the service layer so

[9] Note to reviewers: At the time of the paper submission, the main BioPortal server at `http://bioportal.bioontology.org` runs using the database-based metadata representation. We use the new infrastructure in our development server that is not yet accessible to the public. We expect to transition the new implementation to our production server before the end of the Summer 2009.

that the user interface does not need to be modified and other applications that already use our REST service API can continue to use it.

Since we access the metadata ontology through the Protégé API —which was successfully used in other projects to access in a scalable manner ontologies having more than 8000 classes and 5 million instances [10]— we predict with confidence that the new representation of the metadata about ontologies will scale well. The version of BioPortal currently running on our development server uses this infrastructure successfully, thus validating the approach. We plan to release this implementation to production at end of August 2009.

At the time of this writing, the transition of metadata to ontology-based approach is not complete. Currently, ontology details, ontology versions, and ontology views (see Section 7) are represented as ontology instances. We are in the process of transitioning the rest of the metadata.

7 Validating Extensibility: representing ontology views

As we discussed earlier, one of our main motivations to moving to the ontology-based approach was greater flexibility and adaptability of the metadata. We validated these properties by implementing support for ontology views in BioPortal, which did not previously exist. In our implementation we store only materialized views computed and submitted by the users, together with the metadata describing how the view was generated (the language, engine, etc.)

In this context, a view is any subset of an ontology that is itself an ontology. A view can be created manually or automatically by a view-generation tool. For instance, our collaborators have several views of the Foundational Model of Anatomy (FMA) [15]. One of the views represents the subset of the FMA that would be of interest to a radiologist; another view deals exclusively with Liver; yet another view focuses on the representation of neuroanatomy. The first of these views was generated manually, by starting with the FMA in Protégé and removing the unnecessary branches. The other two views are the results of queries in an extension of SPARQL that our collaborators have developed [17]. When ontology users generate and materialize the views, they often want to share them with other researchers in the field. Thus, when users view a page for FMA, they can see not only its different versions, but also the views available for it.

7.1 Requirements for Representing Views

Discussions with our collaborators led to the following set of requirements on view representation:

- Each view is itself an ontology and can have metadata, be explorable, have reviews, statistics, and so on.
- A view is defined on a specific version of an ontology.

[10] `http://protegewiki.stanford.edu/index.php/Scalability_and_Tuning`

- There is a notion of a "virtual view" (cf. "virtual ontology") such as a view of Liver-related concepts in FMA created for a particular purpose.
  - Each virtual view will have at least one version, but can have several.
  - Each version of the view also has its own metadata (inherited from ontology metadata, but with additional fields).
- We must be able to represent views on views (with the same requirements).
- We must be able to represent views that use more than one ontology.

Note that these requirements suggest a fairly complex structure for view representation, with many cross-references with different components of ontology metadata (ontologies and ontology versions).

7.2 Representing Views in BioPortal

When representing views in BioPortal, we treated them essentially in the same way as regular ontologies, thus getting the browsing, annotation, and other features "for free." We extended the BioPortal Metadata Ontology to represent features that are specific to describing views (Figure 5). A class `VirtualView` is a subclass of `VirtualOntology` and points to the list of versions of the view. Each version of a view is an instance of the class `OntologyView`, which is a subclass of `OMV:Ontology`. Thus, it inherits all the properties that describe an ontology version (e.g., description, domain, author) and adds its own. The view-specific properties include the following:

- The property `viewDefinition` is the textual representation of the view definition. This definition can be a query that was used to create the view, a set of traversal directive (as in Prompt), or any other way that specifies how the view was extracted.
- The property `viewDefinitionLanguage` defines the language that was used to define the view (e.g., the query language, such as SPARQL, for a view that was generated by a query). The range of the property is the class `ViewDefinitionLanguage`, whose instances will have all the pertinent attributes of the language (name, creator, url) as well as the specific version that was used for the view.
- The property `viewGenerationEngine` contains the engine that was used to compute the view. The range of the property is the class `ViewGenerationEngine`, whose instances will have all the pertinent attribute of the engine (name, creator, url) as well as the specific version that was used to generate the view.

7.3 Providing Support for Views in BioPortal

In this ontology-based infrastructure for representing metadata, in order to handle views to BioPortal and satisfying all the requirements that we have outlined earlier, we had to do the following. First, we extended the BioPortal Metadata Ontology as we described in Section 7.2 and Figure 5. After this step, we already had all the structure to represent the views. Second, we implemented the new view-specific REST services (e.g., returning all views for an ontology). This implementation uses the Protégé API to access the view metadata. As we expected, in this implementation we indeed needed to

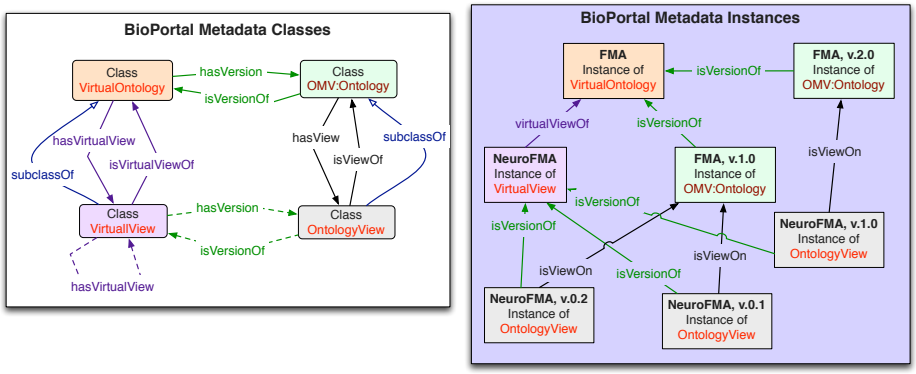

Fig. 5. Classes and instances for representing views: The class `VirtualView` corresponds to the logical description of a specific view (e.g., NeuroFMA, which is an extraction of FMA concepts relevant to Neuroimaging). It is a subclass of `VirtualOntology` because each view is an ontology itself. Similarly, the class `OntologyView` extends the class `OMV:Ontology` and describes specific versions of the view. The right-hand side of the figure shows some example instances of these classes and properties that link them.

focus exclusively on the view-specific logic and structures and did not need to change anything else (e.g., the application infrastructure or the architecture). Some of the functional requirements for handling the views crystallized while we were already working on the implementation, and the only changes that were necessary were the changes to the ontology itself. We did not need to change any database schemas or to add any additional structures (e.g., Hibernate stubs). The work that was required was limited to implementing the logic for the services directly.

In comparison, adding views to the old (Hibernate-based) infrastructure, would have involved some additional changes like: (a) creating new database tables (for views and virtual views), (b) creating a number of foreign keys in order to represent superclass-subclass relationships and relationships expressed by object properties, (c) creating integrity constraints to enforce data integrity along those relations, and (d) generate Hibernate classes (so called "entity beans") from the database tables.

Figure 6 shows the extended ontology-metadata page that includes the information about the views.

8 Discussion and Lessons Learned

The new implementation of BioPortal infrastructure validates the use of ontologies and ontology instances to represent all the metadata as well as system data for a repository of this kind. To the best of our knowledge, BioPortal is the first large-scale web-based application that uses ontologies to represent its internal data. Our experience has shown that using an ontology to describe the knowledge infrastructure of an application does indeed provide the flexibility and adaptability that our application required. Our obser-

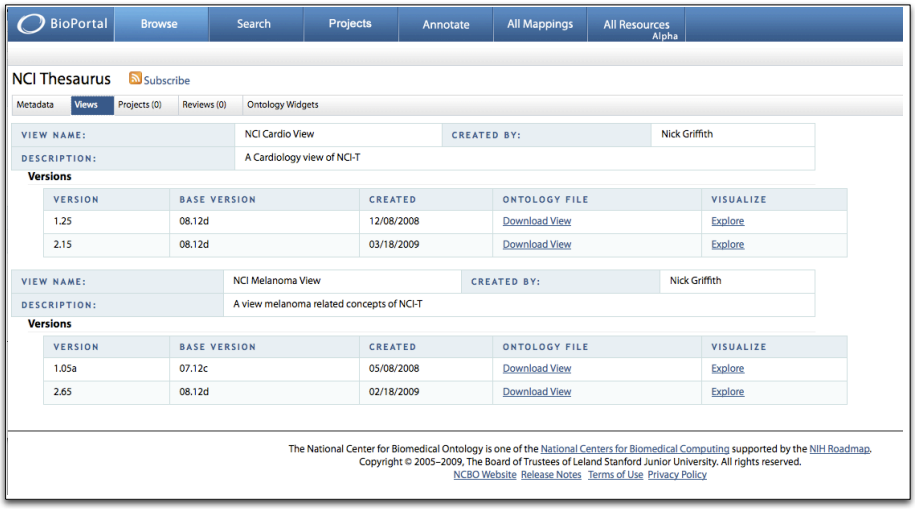

Fig. 6. The user interface for representing view metadata in BioPortal. The screenshot shows two views of the NCI Thesaurus. Each view has two different versions.

vations and discussions with developers show that it was easier and far more efficient to implement view support that satisfied the requirements that we outlined in Section 7.1 using this approach than using a database schema to describe the views and to link them to ontologies. It also enabled us to reuse much of the infrastructure that we have already developed for ontology metadata, as we could treat views as special cases of ontologies. As our requirements for view representation continue to evolve, we know that our changes will be easy to implement as we will need only to evolve the ontology representation and the corresponding service implementations.

It is important to note that some of the design choices in the BioPortal Metadata Ontology are driven purely by application and implementation considerations. Thus, there are parts of this ontology that are specific to the context of BioPortal implementation. We also made some of the ontology-design choices not because they were "ontologically correct" in an abstract sense but because they simplified access to information (cf. `VirtualOntology` class to collect information about all versions of an ontology). Thus, we used the ontology not only as a conceptualization of our domain (metadata representation) but also to represent the physical properties and location of the data.

There are a number of systems that use ontologies to represent some of their metadata. These include Oyster [13], the alignment server [5], and Cupboard [3]. However, in these systems the items in the repository (such as ontologies or alignments) are separate from the part that represents the metadata. Furthermore, the ontology-based metadata does not expand to representing internal system information. The ontology-based representation of metadata focuses on describing intrinsic properties of ontologies and other objects that are shared across applications.

Another class of applications that uses ontologies actively are semantic desktops (e.g., [6]). However, most semantic desktop applications do not represent system data itself using the ontologies. Furthermore, semantic desktop applications focus on creating a "semantic web" of desktop resources rather than representing the internal system data itself.

In our work, we have extended other metadata ontologies. In the future, we plan to integrate other standards. Specifically, we plan to replace the Protégé Mapping ontology that we used here for expediency with an ontology that extends SKOS with mapping-specific metadata.

Our current effort leaves some questions and concerns, however. The main concern is scalability. While Protégé is quite scalable and has been used with ontologies that have hundreds of thousands of classes and instances, we do not know how it will behave with millions of instances describing metadata. As we noted in Section 6, we have not yet moved mappings to this infrastructure. At the same time, we have just uploaded one million new mappings to BioPortal. We are yet to test whether our current Protégé-based infrastructure will be sufficiently scalable for this number of mappings. If we learn that it is not, it will be the limitation of the Protégé implementation itself as modern triplestore implementations easily handle this amount of data. If scalability turns out to be an issue, we will transition to a triplestore to store the instances.

We would like to emphasize that our solution is not limited to the biomedical domain. It so happens that our repository is a repository of biomedical ontologies. However, there is nothing in the BioPortal Metadata Ontology itself or in our use of it that is specific to biomedicine (except perhaps, the list of possible ontology categories). We will install the new infrastructure in other local BioPortal installations, such as the OOR sandbox that we mentioned earlier (and that accepts ontologies in any domain).

Finally, the BioPortal software is open-source. The software is domain-independent and can be used for an ontology repository in any domain or for a domain-independent one. The BioPortal Metadata Ontology is available in BioPortal and can be accessed through the BioPortal user interface or its web services.[11] We have created a snapshot of the instances of the BioPortal Metadata Ontology for our development version of BioPortal. This ontology and instances can be accessed directly through the WebProtégé server at `http://bmir-protege-dev1.stanford.edu/webprotege/`. [12]

Acknowledgments

This work was supported by the National Center for Biomedical Ontology, under roadmap-initiative grant U54 HG004028 from the National Institutes of Health and by NIH grant HL087706. We are grateful to Jim Brinkeley, Todd Detwiller, Onard Mejino, and other members of the Structural Informatics Group at Univesity of Washington for helping us identify requirements for views in BioPortal.

[11] `http://bioportal.bioontology.org`

[12] Please use the Firefox browser to access the server. Select "BioPortal Metadata Ontology" and then go to the "Individuals" tab to view the instances.

References

1. M. V. Bossche, P. Ross, I. MacLarty, B. V. Nu?elen, and N. Pelov. Ontology driven software engineering for real life applications. In *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2007)*, Innsubruk, Austria, 2007.
2. M. Bräuer and H. Lochmann. An ontology for software models and its practical implications for semantic web reasoning. In *The 5th European Semantic Web Conference (ESWC 2008)*, pages 34–48, Tenerife, Spain, 2008. Springer.
3. M. dAquin and H. Lewen. Cupboard a place to expose your ontologies to applications and the community. In *6th European Semantic Web Conference (ESWC 2009)*, Heraklion, Greece, 2009.
4. J. Euzenat. An api for ontology alignment. In *Third International Semantic Web Conference (ISWC 2004)*, pages 698–712, Hiroshima, Japan, 2004. Springer.
5. J. Euzenat. Alignment infrastructure for ontology mediation and other applications. In *Workshop on Mediation in Semantic Web Services*, 2005.
6. T. Groza, S. Handschuh, K. Moeller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjnsdttir. The nepomuk project - on the way to the social semantic desktop. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages 201–211. JUCS, Sept. 2007.
7. L. Hart, P. Emery, R. Colomb, K. Raymond, D. Chang, Y. Ye, E. Kendall, and M. Dutra. Usage scenarios and goals for ontology definition metamodel. In *5th International Conference on Web Information Systems Engineering (WISE 2004)*, pages 596–607, Brisbane, Australia, 2004. Springer.
8. T. Lehmann. A framework for ontology based integration of structured it-systems. In *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2007)*, Innsubruk, Austria, 2007.
9. N. F. Noy, A. Chugh, W. Liu, and M. A. Musen. A framework for ontology evolution in collaborative environments. In *Fifth International Semantic Web Conference, ISWC*, volume LNCS 4273, Athens, GA, 2006. Springer.
10. N. F. Noy, N. Griffith, and M. A. Musen. Collecting community-based mappings in an ontology repository. In *7th International Semantic Web Conference (ISWC 2008)*, Karlsruhe, Germany, 2008.
11. N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute, and M. A. Musen. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 2009.
12. C. I. Nyulas, M. J. O'Connor, S. W. Tu, A. Okhmatovskaia, D. Buckeridge, and M. A. Musen. An ontology-driven framework for deploying jade agent systems. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Sydney, Australia, 2008.
13. R. Palma, P. Haase, and A. Gomez-Perez. Oyster: sharing and re-using ontologies in a peer-to-peer community. In *15th international conference on World Wide Web (WWW 2006)*, pages 1009–1010, Edinburgh, Scotland, 2006. ACM.
14. R. Palma, J. Hartmann, and P. Haase. OMV: Ontology Metadata Vocabulary for the Semantic Web. Technical report, http://ontoware.org/projects/omv/, 2008.
15. C. Rosse and J. L. V. Mejino. A reference ontology for bioinformatics: The Foundational Model of Anatomy. *Journal of Biomedical Informatics.*, 2004.
16. H. H. Shahri, J. A. Hendler, and A. A. Porter. Software configuration management using ontologies. In *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2007)*, Innsubruk, Austria, 2007.
17. M. Shaw, L. T. Detwiler, J. F. Brinkley, and D. Suciu. Generating application ontologies from reference ontologies. In *AMIA Annual Symposium*, Washington, DC, 2008.