An Ontology-driven Architecture for Extensible Scientific Data Management Systems

Yuan-Fang Li*, Gavin Kennedy, Faith Davies, Jane Hunter

School of ITEE, The University of Queensland
GP South, Staff House Road, St Lucia 4072, Australia

Abstract

Data management has become a critical challenge faced by a wide array of scientific disciplines in which the provision of sound data management is pivotal to the success and impact of research projects. The huge and rapidly growing amounts of data to be managed and the fact that the *models* of data evolve over time contribute to making data management an increasingly complex undertaking that warrants a rethinking of its design. We believe that a number of intrinsic characteristics of Semantic Web ontology languages OWL and RDFS, such as semantic rigor and the extensible nature, make them an ideal conceptual platform on which effective data management systems can be developed. In this paper, we present our efforts in designing an open and extensible architecture with an ontology at its core. In this architecture, the behaviors of domain concepts and objects are captured entirely by ontological entities, around which all data management tasks are carried out. Moreover, the open and semantic nature of ontology languages also makes such systems amenable to greater data reuse and interoperability. To the best of our knowledge, this is the first proposal of an ontology-driven architecture for extensible data management systems. An ideal domain for applying these principles is phenomics, the systematic study of phenotypes of organisms. Phenomics research generates high volumes of heterogeneous data and makes use of emerging imaging and measurement technologies and processes, thus making it an ideal testbed for data management systems. In this context, we describe the development of PODD, a data management system for phenomics research, as a step towards validating the practicality of the ontology-driven architecture.

Key words: Ontology-driven architecture, OWL, data management systems, PODD

*Author for correspondence

Email addresses: `uqyli4@uq.edu.au` (Yuan-Fang Li), `g.kennedy1@uq.edu.au` (Gavin Kennedy), `f.davies@uq.edu.au` (Faith Davies), `j.hunter1@uq.edu.au` (Jane Hunter)

1. Introduction and Background

Data management is the practice of managing (digital) data and resources, encompassing a wide range of activities including acquisition, storage, retrieval, discovery, access control, publication, integration, curation and archival. For many data-intensive scientific disciplines such as life sciences and bioinformatics, sound data management informs and enables research and it has become an indispensable component [1].

The need for effective data management is, in a large part, due to the fact that huge amounts of digital data are being generated by modern instruments. Furthermore, the fast evolution of technologies/processes and discovery of new scientific knowledge require flexibility in handling dynamic data and models in data management systems.

Database systems have traditionally been used successfully to manage research data [2]. Database schemas are used as domain models to capture the attributes and relationships of domain concepts. One implication of the above approach is that domain models need to stay relatively stable as database extension or migration is often an error-prone and laborious task in practice. Hence, we believe that this approach is not suitable for domains where data and model evolution is the norm rather than the exception.

Semantic Web ontology languages such as RDF Schema [3] and OWL [4] possess expressive, rigorously-defined semantics and non-ambiguous syntaxes. Moreover, they have been designed to be open and extensible to support knowledge and data exchange on the Web scale [5, 6]. These intrinsic characteristics make them an ideal conceptual platform on which a flexible data management system that supports dynamic data and models can be built.

Ontology language OWL has been widely used in a number of domains, notably in life sciences and biotechnology [7, 8, 9] as a modeling language for its expressivity and extensibility. There is also growing tool support for tasks such as reasoning, querying and visualization, making it a viable option for the modeling and representation of scientific domain concepts.

Moreover, with the rapid progression of Semantic Web-based data integration through the community-driven Linked Data project [10], it is advantageous for data management systems to support Semantic Web languages and standards natively to benefit from the fast-increasing, integrated open datasets.

In this paper, we present our work in designing a domain-independent, ontology-driven architecture for scientific data management systems. In our architecture, we support data and model dynamics through ontology-based domain modeling. In this architecture, the ontology is at the core of the system where the behaviors of abstract domain concepts and concrete domain objects are entirely defined with ontological vocabularies. Logical structure of data is therefore maintained and enforced via ontological definitions and reasoning and not via database schemas and associated constraints.

We also describe the modeling of the base ontology in this architecture and show how the domain can be conceptualized in an ontology in an extensible and flexible way.

Based on such a design, we have developed the Phenomics Ontology Driven Data (PODD) data management system [11] that supports the effective management of large-scale, evolving data for phenomics research. An example of concept change is the concept *GrowthCondition* in phenomics, which captures the environmental conditions in which an organism grows. For different disciplines and projects, it can be specialized to *CabinetCondition* to capture conditions in controlled environments and *FieldCondition* to capture conditions in the field.

The ontology-based domain model is at the core of PODD as it drives the creation, storage, validation, query and search of data and metadata. In contrast to traditional data management systems that use database schemas as the underlying model, the employment of OWL ontologies as the domain model makes PODD highly extensible.

In biology, an organism's phenotype is its observable or quantifiable trait, such as the height of a plant or the size of a leaf, as a consequence of its genetic makeup combined with its developmental stage, environment and disease conditions. Phenomics is the systematic and comprehensive study of an organism's phenotype and is determined through a combination of high-throughput and high-resolution imaging- and measurement-based analysis platforms. Phenomics research, together with genomics research, represents a holistic and promising new approach to biological study [12, 13, 14, 15].

Unlike genomics, phenomics research emphasizes the study of physical, observable traits of organisms. Like genomics, vast amounts of data are produced by imaging and measurement platforms and analysis tools. The effective storage, management, analysis and discovery of these data is a challenging problem. Specifically, there are three key challenges for data management in phenomics research.

- The ability to provide a data management service that can manage large quantities of heterogeneous data in multiple formats (text, image, video) and not be constrained to a finite set of imaging and measurement platforms and data formats.
- The ability to support metadata-related services to provide context and structure for data within the data management service to facilitate effective search, query and dissemination.
- The ability to accommodate evolving and emerging technologies and processes as phenomics is still a rapidly developing field of research.

PODD has been developed to meet the above challenges facing the Australian phenomics research community by providing efficient and flexible repository functionalities for large-scale phenomics data, and providing a mechanism for maintaining structured and precise *metadata* around the raw data so that they can be stored, distributed and published in a reusable fashion. We would like to emphasize that although the PODD system is geared towards phenomics research, the ontology-driven architecture we propose in this system is actually domain-independent and can be applied in any scientific discipline where research output can be conceptually organized in a structured manner.

The rest of the paper is organized as follows. In Section 2 we present related work and give a brief overview of the motivation and goals of the PODD project. Section 3 presents in detail the ontology-based architecture for data management systems. In Section 4, in a bioinformatics setting, we discuss the PODD domain ontology in more detail and show how the ontology-based modeling approach is used in the life cycle of domain objects. In Section 5, we describe the implementation of the PODD data management system. Finally, Section 6 concludes the paper and identifies future directions.

2. Related Work, Motivation & Goals

Over the years attempts have been made to develop content repository systems and architectures to meet institutional and personal data management needs. In this section, we introduce a number of such systems and architectures. With a survey of related work, we present the motivation behind the ontology-driven architecture and the goals we wish to achieve with the PODD data management system.

2.1. Data and Resource Management Systems

A number of open-source content repository specifications and software systems have been developed.

Fedora Commons[1] is an open-source digital resource management system based on the principles of modularity, interoperability and extensibility. In Fedora Commons, abstract concepts are defined as *models*, on which inter-relationships and behaviors can be further defined. Data in Fedora Commons repositories are organized into *objects*, which have *datastreams* that stores either metadata or data. Fedora Commons makes heavy use of Semantic Web technologies through the use of common RDF vocabularies and the integration with the Mulgara triple store[2], which can be used for metadata storage and query through SPARQL. Fedora Commons has been used as a backend to implement document management systems, digital libraries and institutional repositories.

Apache Jackrabbit[3] is an open-source implementation of the Content Repository for Java Technology (JCR) API[4]. In JCR, data is stored in a tree of *nodes*, which can hold *properties* of arbitrary values, which is conceptually similar to Fedora Commons. Types can be defined on nodes to place certain restrictions on them.

[1] `http://www.fedora-commons.org/`
[2] `http://www.mulgara.org/`
[3] `http://jackrabbit.apache.org/`
[4] `http://jcp.org/en/jsr/detail?id=283`

Fedora Commons and JCR both support fairly basic mechanisms for defining object relationships. Hence, they are usually used as the underlying repository solution on which complex data and document management systems are built. These systems include Biodiversity Heritage Library[5] and Fez[6], among others. As stated previously, these systems use database schemas as their domain models.

Data management systems have also been developed to support a number of scientific disciplines including high-energy physics, bioinformatics and Earth observation.

Bioinformatics Resource Manager (BRM) [2] is one example of client-server style data management software for bioinformatics research. The client software is installed on users' computers to access (microarray and proteomic) resources stored on BRM server in a PostgreSQL relational database. The BRM server supports data acquisition from external sources such as NCBI [16] and UniProt [17]. It also supports annotation using public datasets and connectivity to analytics tools. Data in BRM is stored under the *Project* concept and is mostly flat, i.e., it does not support hierarchical domain concepts such as investigation and publication.

Besides data management systems, grid-based middleware systems have also been developed to provide distributed storage solutions. Such systems include the Storage Resource Broker (SRB) [18] and the CERN Data Grid [19] and other systems that make use of Globus[7] middleware. These systems store data in a distributed environment and usually support authentication, replication, redundancy, etc. However, they are mostly only concerned about data storage and replication and hence do not provide full-fledged data management capabilities. Interested readers can see [20] for a detailed survey of grid resource management systems.

2.2. Domain Modeling in Scientific Research

A number of specifications and ontologies have been proposed to model scientific research activities. In 2004, Council for the Central Laboratory of the Research Councils (CCLRC) of UK developed a CCLRC Scientific Metadata Model [21] that models scientific activities in free text. An OWL ontology, EXPO, was developed [22] to capture metadata about scientific experiments. EXPO was developed in a top-down approach by extending concepts in the Suggested Upper Merged Ontology (SUMO)[8]. Although very comprehensive, these models are very verbose and not very suitable as a model for developing data management systems.

In biological and particularly 'omics research, a large number of databases have been developed to host a variety of information such as genes (Ensembl[9]),

[5]`http://www.biodiversitylibrary.org/`
[6]`http://fez.library.uq.edu.au/`
[7]`http://www.globus.org/`
[8]`http://www.ontologyportal.org/`
[9]`http://www.ensembl.org/`

proteins (UniProt[10]), scientific publications (PubMed[11]) and microarray data (GEO[12]). These databases are generally characterized by the fact that they specialize in a particular kind of data (protein sequences, publications, etc.) and their conceptual domain models, such as gene and gene products [9] and microarray experiments [23], are well understood.

For a scientific data management system to be effective, models of domain concepts need to be integrated with models of scientific activities and workflows. However, models of biological and clinical investigations are less well understood.

The Ontology for Biomedical Investigations (OBI)[13] is an ongoing effort of developing an integrative ontology for biological and clinical investigations. It takes a top-down approach by reusing high-level, abstract concepts from other ontologies. It includes 2,600+ OWL [4] classes and 10,000+ axioms (in the import closure of the OBI ontology). Although OBI is very comprehensive, its size and complexity makes reasoning and querying of OBI-based ontologies and RDF graphs computationally expensive and time consuming, making it impractical as a domain model for a data management system where such reasoning may need to be performed repeatedly.

Functional Genomics Experiment Model (FuGe) [24] is an extensible modeling framework for high-throughput functional genomics experiments, aiming at increasing the consistency and efficiency of experimental data modeling for the molecular biology research community. Centered around the concept of experiments, it encompasses domain concepts such as protocols, samples and data. FuGe is developed using UML from which XML Schemas and database definitions are derived. The FuGe model covers not only biology-specific information such as molecules, data and investigation, it also defines commonly used concepts such as audit, reference and measurement. Extensions in FuGe are defined using inheritance of UML classes.

We feel that the extensibility we require is not met by FuGe as any addition of new concepts would require the development of new database schemas and code. Moreover, the concrete objects reside in relational databases, making subsequent integration and dissemination more difficult.

2.3. The PODD Data Management System – Motivation & Goals

Phenomics is a fast-growing, data-intensive discipline with new technologies and processes rapidly emerging and evolving. As a result, its domain model and data management systems must also be able to evolve to handle the complexity, change and scale.

The PODD system is being developed on behalf of two major phenomics initiatives: the Australian Plant Phenomics Facility (APPF), specializing in phenotyping crop and model plant species; and the Australian Phenomics Network (APN), specializing in the phenotyping of mouse models. Both facilities

[10]`http://www.uniprot.org/`
[11]`http://www.ncbi.nlm.nih.gov/pubmed/`
[12]`http://www.ncbi.nlm.nih.gov/geo/`
[13]`http://purl.obolibrary.org/obo/obi`

have common requirements to gather and annotate data from both high- and low-throughput phenotyping devices. The scale of measurement can be from the micro or cellular level, through the level of a single organism, and up to (in the case of the APPF) the macro or field level. Imaging, measurement and analysis of organisms on such a large scale will produce an enormous amount of data. For example, it has been estimated that more than 60 TB of image data will be generated by APPF research centers every year, which need to be managed effectively.

Phenomics research makes use of a large variety of imaging and measurement platforms. For example, in mouse histopathology and organ pathology research, the Zeiss "Mirax Scan" scanner is used to scan microscope slides. In clinical pathology, a Flow Cytometer is used to capture laser diffraction images of blood samples. In plant research, the Lemnatec Scanalyzer is used to capture RGB images of plants in growth cabinets. The Fluorogroscan system is used in quenching analysis: the partitioning of light energy used in photosynthesis on model plants such as Arabidopsis. Other devices, such as Infrared Thermography Camera to capture leaf temperature and SPAD Meter to measure the chlorophyll content of plant leaves, are also used. New devices and instruments will also be used when they become available. Moreover, existing instruments may be upgraded so that they can capture more information. The PODD domain model needs to be flexible to accommodate these changes.

Because an organism's phenotype is often the product of the organism's genetic makeup, combined with its development stage, disease conditions and its environment, any measurement made against an organism needs to be recorded in the context of these other *metadata*. Consequently the opportunity exists to create a repository to record the data, its contextual data (metadata) and data classifiers in the form of ontological or structured vocabulary terms. The structured nature of this repository would support manual and autonomous data discovery as well as provide the infrastructure for data based collaborations with domestic and international research institutions. Currently there are no such integrated systems available to the two facilities. The National eResearch Architecture Taskforce (NeAT) Australia initiated the PODD project to fill this gap. In PODD, we have engaged in the design and development of the Phenomics Ontology Driven Data (PODD) repository. The goals of PODD are to capture, manage, annotate and distribute the data generated by mouse and plant phenomics research activities. It supports both Australian and international biological research communities by providing repository and data publication services.

3. The Architecture of the Ontology-driven Data Management System

3.1. Requirements of Data Management Systems

For any scientific data management systems, a number of requirements need to be satisfied.

Data storage and management Research activities in data-intensive disciplines such as 'omics often generate huge amounts of data. The ability to efficiently acquire, store and manage large volumes of data is essential.

Data contextualization Sufficient contextual information needs to be maintained for more effective organization, understanding and discovery of raw data. Contextual information includes both conceptual domain models, such as how research activities are organized and carried out; and metadata such as provenance information.

Data security There are many dimension to data security, including access control, versioning and backup. An effective data management system needs to ensure data security through the use of authentication and authorization and sound versioning and backup solutions.

Data identification and longevity In order to support the dissemination of scientific findings, data in the repository needs to be publicly accessible after being published. Hence, a persistent and unique naming scheme is required. Moreover, valuable scientific data also need to be stored in perpetuity.

Data reuse and integration Contextual information helps to make sense of raw data. Moreover, it also needs to be made discoverable, through means such as full-text search, faceted browsing and complex query answering, to allow raw data to be integrated and reused.

Model extensibility A data management system may need to manage a wide variety of data, which may be generated by different software and captured by different platforms. An expressive and extensible domain model is therefore essential to cater for modification, addition and deletion of domain concepts. The data management system also needs to be designed to minimize service disruption when such a model change occurs.

3.2. The Ontology-driven Architecture For Data Management Systems

The most distinguishing characteristic of the ontology-driven architecture is the central role ontologies play. In this architecture, raw data are not stored in a flat structure but are attached in domain objects organized in a logical, hierarchical system, defined according to the domain model that represents the structure of research activities.

Current document management systems such as Fez[14] typically have a relatively static domain model and hardwire it as relational schemas and foreign-key constraints in a custom relational database independent from the underlying repository system. Consequently, the information pertinent to the model of each concrete object is stored in this custom database as well. As stated in the

[14]`http://fez.library.uq.edu.au/`

previous section, this approach is unsuitable for dynamic environments where conceptual changes are common.

To effectively support a dynamic conceptual framework, the domain model in the proposed architecture is defined using OWL ontologies, in which: OWL classes represent domain concepts; OWL properties define concept attributes and their relationships; OWL restrictions specify constraints on concepts and finally; OWL individuals define concrete domain objects where attributes and relationships are defined using OWL assertions. Raw data files are attached to concrete domain objects.

Such a conceptual architecture alleviates the problem of imposing hard relational constraints in a database which is difficult to extend/change. It is worth noting that referential integrity is not sacrificed in achieving flexibility: ontological reasoning involving relevant concepts and objects are performed before object modification to ensure that all constraints are satisfied.

Another drawback of existing systems is that there can be only one domain model. When a concept needs to be updated, all the existing objects need to be updated accordingly, which may be undesirable, inappropriate and time-consuming. This is, unfortunately, unavoidable as long as the domain model is defined using database schemas. In our proposed architecture, as concept and object definitions are stored in the repository, such changes can be versioned so that existing instance objects can remain legitimate when integrity validation is performed as they can still refer to the previous conceptual definitions.

Similarly, the data and metadata (including ontological definitions) of each object can be modified, and the modifications should be versioned so that they can be rolled back.

The major differences between the ontology-driven architecture and that of current systems can be summarized in Table 1 below.

Table 1: Summary of major conceptual differences between the ontology-driven architecture and current systems.

Aspect	**PODD**	**Current systems**
Domain modeling	OWL Ontologies	Database schemas
Model organization	In repository, using OWL classes, properties & individuals	In custom databases, hardwired
Object organization	Ontological definition an integral part, stored in the repository	Stored in databases
Model change	Creates new version of OWL definitions, old objects unaffected	Database & schema migration, old objects need to be updated
Object change	Creates new version of OWL definitions	Table updates
Referential integrity	OWL reasoning	Database integrity check

In developing the ontology-driven architecture, the following design decisions have been made to balance expressivity, flexibility and conceptual clarity.

- There is a top-level domain concept, called *Project* [15], under which other concepts (such as *Investigation* and *Material*) reside in a hierarchical manner.
- Access control (authorization) is defined on the *Project* level but not on an individual object level, i.e., a given user will have the same access rights for all objects within a given project.
- Within a *Project* hierarchy, objects are in a parent-child relationship in a tree structure such that each child can only have one parent. This ensures that access rights are properly propagated from parent to child and there is no chance of confusion.
- Additionally, inter-object, many-to-many reference relationships can be defined to enhance flexibility of the architecture as it allows arbitrary links between objects to be established.
- Objects cannot be shared across *Projects*. Instead, objects must be copied from one project and pasted into another one. Such a rule simplifies object management with the elimination of possible side-effects caused by sharing object between projects.
- There should be no interference between different versions of a given concept and between objects that are instances of different concept versions.

The high-level design of ontology-driven architecture takes a modular and layered approach, as can be seen in Figure 1. At the foundation is the **data access layer**, consisting of an underlying repository system, an RDF triple store, an in-house database that stores essential information and a full-text search engine. This layer is responsible for low-level tasks when the creation, modification and deletion of concepts and objects occur. The **business logic layer** in the middle is responsible for managing concepts and objects, such as versioning, object conversion and integrity validation. The **security layer** controls access (authentication and authorization) to concepts and objects and guards all operations on them. In this architecture, authorization is based on user attributes, which have two dimensions. Firstly, each user has a system-wide role, such as registered user or system administrator, which is used to determine access rights across the system. Secondly, a project-wide role, such as project administrator and project observer, can be assigned to a user so that he can have project-specific access rights. At the top of the stack is the **interface layer**, where the data management system can be accessed using a number of interfaces such as a Web browser or API calls.

[15] The choice of concept names in the domain ontology is actually irrelevant to the proposed architecture. Names such as *Project* and others are chosen as they are general and representative enough.

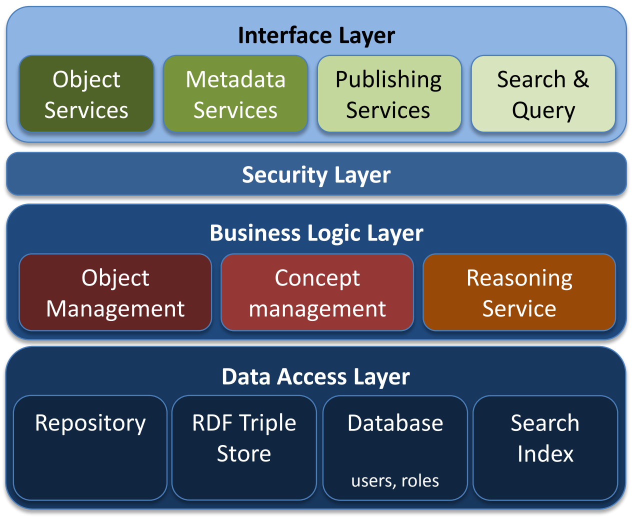


Figure 1: A high-level depiction of components in the ontology-driven architecture.

4. Ontology-based Domain Modeling

As we emphasized previously, the domain model should be flexible enough to accommodate the rapid changes and dynamic nature of scientific research. In this section, we present the base ontology and the roles it plays in the ontology-driven architecture. It should be noted that the architecture proposed here is domain-independent and it be applied to any scientific discipline that shares a similar high-level domain model.

4.1. The Base Domain Ontology

Inspired by FuGe and OBI, we create the base domain ontology in OWL to define essential domain concepts, their attributes and inter-relationships in an object-oriented fashion. As stated in the previous section, domain concepts will be modeled as OWL classes; relationships between concepts and object attributes will be modeled as OWL object and datatype properties. Concrete objects will be modeled as OWL individuals.

For an overview, attributes and inter-relationships of some of the domain concepts and the core metadata of all concepts in this ontology are shown in Figure 2.

First of all, we set out a few design principles of the domain ontology.

- All essential domain concepts are modeled as subclasses of an abstract top-level OWL class ***PODDConcept*** that captures common attributes and relationships.
- All relationships between domain concepts are captured by *domain properties*, which can be further divided into two *property hierarchies*, one for

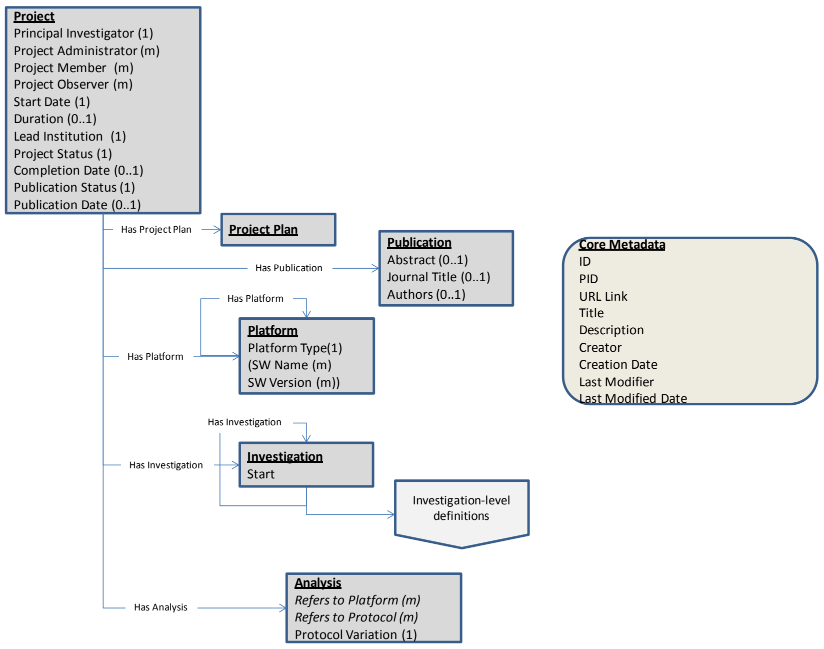


Figure 2: The attributes and relationships of some domain concepts.

parent-child relationships and the other for reference relationships. Each of the two hierarchies have an abstract top-level property, called ***contains*** and ***refersTo***, respectively.

- All parent-child relationships are modeled in a property hierarchy as subproperties of the abstract property ***contains***, and all reference relationships are modeled in another property hierarchy as subproperties of the abstract property ***refersTo***.

- For each domain concept T, one property is defined in each of the above hierarchies with its range defined to be T. The domains of such properties are not specified so that they can be used by any applicable domain concept to establish a relationship between them.

- Domain concepts (except ***PODDConcept***) are mutually disjoint with each other.

- Essential domain concepts can be subclassed to provide more specialized and refined information.

Moreover, to ensure that each object can have at most one parent object, the inverse property of ***contains***, ***isContainedBy***, is defined so that a max cardinality restriction can be added to the top-level concept ***PODDConcept*** to enforce it.

The definitions of the high-level constructs can be summarized in Figure 3, in OWL DL syntax [4].

$$PODDConcept \sqsubseteq \top \qquad isContainedBy = (^{-}contains)$$
$$\top \sqsubseteq \forall contains.PODDConcept \qquad \top \sqsubseteq \forall refersTo.PODDConcept$$
$$PODDConcept \sqsubseteq \leq 1\ isContainedBy$$

Figure 3: Top-level ontology constructs in the PODD ontology.

4.1.1. The Domain Concepts

In the base domain ontology, essential domain classes include the following:

Project is the top-level concept (under ***PODDConcept***) representing an administrative concept that contains essential meta information about the research project, such as the administering organization, principal investigator, project membership, project status, etc.

ProjectPlan describing the research plan of the project at the core metadata element level.

Platform describing any single technical measurement platform used in the project. Technical measurement platform means any platform for which parameters and parameter values may be captured.

Investigation describing a planned process within a project that executes some form of study design and produces a coherent set of results. It can be considered equivalent to an experiment.

The ***Investigation*** concept is of central importance. It captures the data and metadata of experiments under a project. A number of concepts are defined to assist in the modeling of investigations.

Design describing experimental design components, e.g. plant layouts, sampling strategies, etc.

Process representing a planned component of an investigation. It is a description of a series of steps taken to achieve the objective of the investigation.

Protocol describing a step within a process that is a consistent whole. e.g. sterilize seeds, plant seedlings, image the plants.

Material describing the materials used in the investigation. Materials can be either inputs or outputs. They can be chemicals, substrates, whole organisms or samples taken from a whole organism. The meaning of the *Material* is usually dependent on the domain and hence can be specified by domain-specific properties.

Event capturing ad-hoc events and actions that occur against an individual material. In most instances the events and their timing are described in the process/protocol. An *Event* object can be utilized to either record fixed events in a form that allows for investigative analysis, or to record one off observations (e.g., the plant under observation died).

Measurement describing a single measurement against a single material. e.g. an image of a plant is a single measurement. *Measurement* objects can capture measurement variables (e.g. shutter speed, lighting, etc).

Analysis is a metadata concepts used to describe outputs derived from measurements. An analysis can be performed against a single material object, a set of investigations, or at the level of the whole project.

4.1.2. Inter-concept Relationships

The structures and workflows of phenomics research activities are captured using OWL object properties.

Following the design principles presented in the beginning of this section, domain properties such as ***hasAnalysis*** and ***refersToPlatform*** can be defined as follows in Figure 4.

$$\begin{array}{ll} \mathit{hasAnalysis} \sqsubseteq \mathit{contains} & refersToPlatform \sqsubseteq refersTo \\ \top \sqsubseteq \forall \mathit{hasAnalysis}.\mathit{Analysis} & \top \sqsubseteq \forall refersToPlatform.Platform \end{array}$$

Figure 4: Example definitions of domain properties

4.1.3. Object Attributes

Attributes are intrinsic properties of an object, such as the status of a project, the start date of an investigation and the timestamp of an event. In our model, we use OWL properties to model object attributes, similar to the modeling of inter-object relationships.

When the possible values of a particular attribute can be enumerated, such as project status (*active*, *inactive* and *completed*), an enumerated OWL class is used to represent all the values. When an attribute represents a grouping of some values, such as accessions, where an accession has a source and a number, an OWL class is also defined to represent the grouping. In this case, auxiliary OWL properties are defined to project out specific values in the grouping. In all other cases, attributes are modeled using datatypes.

Figure 5 shows the partial definition of the OWL class *Project*. Restriction (1), for example, states that any *Project* instance must have exactly one *ProjectPlan* (through the predicate *hasProjectPlan*, the range of which is *ProjectPlan*). The other 3 restrictions are similarly defined.

4.2. Roles of Domain Ontologies in Object Life Cycle

The base ontology defines essential concepts independent of the domain. Domain-specific information can be then captured by extending the base ontology in individual systems.

$$
\begin{aligned}
\mathit{Project} \sqsubseteq & \ = 1\ \mathit{hasProjectPlan} && (1)\\
\sqsubseteq & \ \geq 1\ \mathit{hasInvestigation} && (2)\\
\sqsubseteq & \ = 1\ \mathit{hasStartDate} && (3)\\
\sqsubseteq & \ \leq 1\ \mathit{hasPublicationDate} && (4)
\end{aligned}
$$

Figure 5: Partial OWL Definition for the Project concept.

Concrete objects which are instantiations of various concepts such as ***Project*** and ***Investigation***, are stored in the repository and can subsequently be retrieved for different purposes. As stated in Section 1, the ontology-based domain model is at the center of the whole life cycle of objects. In this subsection, we briefly describe the roles the domain ontologies perform at various stages of the object life cycle.

Ingestion When an object is created, the user specifies which type of object she intends to create and the repository will pull up all the ontological definitions for that type (from the OWL class corresponding to that type and its super classes). Such definitions will be used to (a) guide the rendering of object creation interfaces and (b) validate the attributes and inter-object relationships the user has entered before the object is ingested. When the object is ingested, its metadata definitions are stored as RDF assertions.

Retrieval & update When an object is retrieved from the repository, its attributes and inter-object relations are retrieved from its RDF assertions, which are used to drive the on-screen rendering. When any value is updated, it is validated and updated in this object's RDF assertions.

Query & search An object's assertions will be stored in an RDF [25] triple store, which can be queried using the SPARQL [26] query language. Similarly, ontology definitions are indexed to provide functionalities such as full-text search and faceted browsing.

Publication & export When an object is published or exported, its metadata, in RDF, will be retrieved and exported.

4.3. Integration with Existing Domain Ontologies

As stated before, ontologies such as Gene Ontology [9] and Plant Ontology [28] are widely used in biomedical research to capture information such as genes, proteins, sequences and organism phenotypes. In our ontology-driven approach, these ontologies will be used to add annotations on domain objects to enrich their semantic descriptions and enable cross-application reference and integration.

In summary, ontology-based domain modeling enables us to build very expressive and extensible conceptual models which can be extended to accommodate individual domains. Ample tool support is also available to perform ontology-based tasks such as validation, query answering and searching.

5. The PODD Data Management System

Based on the ontology-driven architecture presented in Section 3 and the domain ontology presented in Section 4, we have developed the PODD data management system to meet the data management challenges faced by the Australian phenomics research community.

To describe domain knowledge in phenomics, we extend the base ontology described in Section 4 by defining the following concepts: *Genotype*, *Gene*, *Phenotype* and *Sequence* as subclasses of *PODDConcept*. Additional OWL object and datatype properties are also defined to model the attributes and relationships of these concepts, such as shown in Figure 6. Note that the last two definitions integrate the new definitions with those in the base ontology.

$$\begin{aligned} Genotype &\sqsubseteq PODDConcept \\ &\sqsubseteq \forall hasGene.Gene \\ &\sqsubseteq \;\leq 1\ hasEcotype \\ &\sqsubseteq \;\leq 1\ hasSubspecies \\ &\ldots \end{aligned} \qquad \begin{aligned} Gene &\sqsubseteq PODDConcept \\ &\sqsubseteq \forall hasSequence.Sequence \\ &\sqsubseteq \;\leq 1\ hasAlias \\ &\sqsubseteq \;\leq 1\ hasChromosome \\ &\ldots \end{aligned}$$

$$\begin{aligned} Phenotype &\sqsubseteq PODDConcept \\ &\sqsubseteq \forall hasMeasurement.Measurement \\ &\ldots \end{aligned}$$

$$\begin{aligned} Project &\sqsubseteq \forall hasGenotype.Genotype \\ Material &\sqsubseteq \forall hasPhenotype.Phenotype \\ &\sqsubseteq \forall refersToGenotype.Genotype \end{aligned}$$

Figure 6: Domain-specific OWL defintions.

The above definitions show how the new OWL classes are integrated with existing ones through the use of OWL object properties. Conceptually, it states that phenotype objects are attached to material objects, which in turn can reference genotype objects, which are defined as child objects of projects.

As can be seen above, the domain model is dynamic in that it accommodates the addition of new classes and properties. As a result, the other components of the PODD system is also designed to be amenable to the dynamic nature of the architecture. The PODD system is developed in Java for its cross-platform availability and the huge ecosystem around it. A more detailed architecture diagram for PODD can be seen in Figure 7. Below we summarize a number of core technologies employed in the development of the PODD data management system and describe briefly how we use them to achieve extensibility and scability.

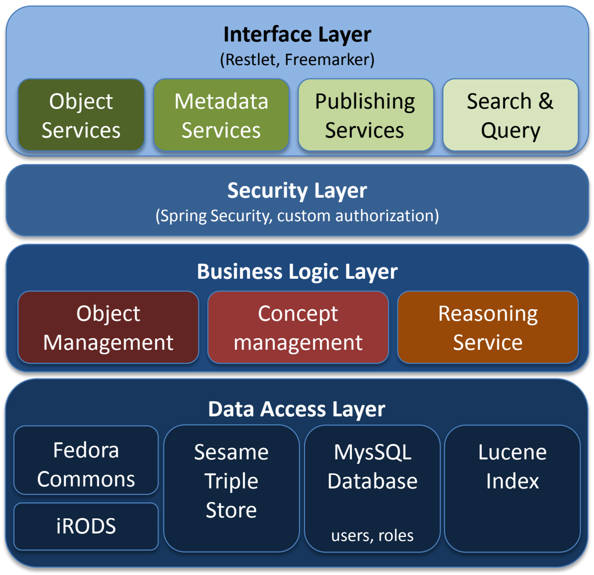


Figure 7: The architecture and components of PODD.

- We use Fedora Commons, a digital repository for the storage and retrieval of domain objects.

  As introduced in Section 2, Fedora Commons is a mature, widely-used digital repository software. In PODD, we store domain concepts as Fedora models and concrete domain objects as Fedora objects. The OWL (for concepts) and RDF (for objects) definition of each concept and object is stored in a special versioned datastream `PODD`, which is used by the system in various tasks such as object creation, rendering, validation and update. Moreover, raw data files are also stored in Fedora Commons as versioned datastreams.

  It is to be noted that the structure of domain objects is not maintained in Fedora Commons but rather in PODD system on top of it. In other words, Fedora Commons is not aware of the semantics of concepts and objects.

- We use iRODS [27], a distributed, grid-based storage software system, as the storage module for Fedora Commons to provide a distributed storage solution. Transparent to Fedora Commons, iRODS manages the distribution and replication across a virtual, geographically-distributed data fabric.

  iRODS provides two important benefits: (1) data transfer can enjoy principle of locality by replicating files at strategically placed data storage nodes in the fabric; and (2) storage size is theoretically unlimited as the data fabric can grow as demand grows.

- We incorporate the Sesame[16] triple store to support complex query answering with SPARQL. It also provides metadata to the full-text index and search component.

  For each object, its RDF definitions are stored in a Sesame *context* that uniquely scopes and identifies the triples. The use of contexts is necessitated by two facts. Firstly, it cannot be guaranteed that triples are unique across all objects. For example, *Accession* objects are defined as RDF blank nodes with an accession source and an accession number. Hence, it is possible that two objects contain the *same* accession object, which cannot be distinguished in the triple store. Hence, contexts are necessary to identify all triples of a given object. Secondly, as described in Section 3, access control needs to be enforced on a project level. Similarly, it also needs to be enforced on query answering in the triple store. By identifying triples of individual objects, we are able to control contexts a user can access through query expansion.

- We use the Solr[17] open-source search engine platform to provide full-text search and faceted browsing capabilities of repository contents, including values in the RDF triple store.

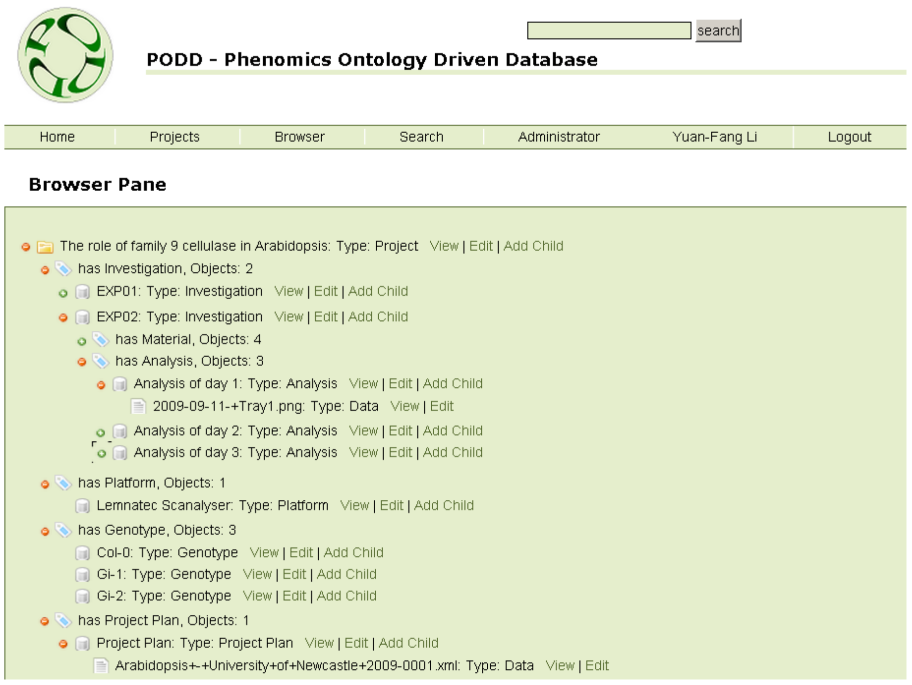
  Similar to the structure of the Sesame triple store, there is a one-to-one correspondence between domain objects in the repository and the Solr *documents*, which represent logical indexing units. Moreover, through search query expansion, access control can be applied to search results.

- We use MySQL database to store user and access control related information, as it is orthogonal to other domain concepts.

Figure 8 shows the browser view of a plant phenomics project that investigates Arabidopsis. Note that in this view, the objects are shown in a tree-like structure by following the property assertions of subproperties of *contains* defined in the base and domain ontologies.

Following the ontology-driven architecture and through the use of mature technologies, we have successfully developed PODD, a scalable, extensible data management system. At the same time, data management tasks such as versioning, logical organization of data, authentication & authorization, and discovery, can all be supported effectively. In summary, the PODD system demonstrates the feasibility and practicality of the proposed ontology-driven architecture.

[16]`http://www.openrdf.org/`
[17]`http://lucene.apache.org/solr/`



The role of family 9 cellulase in Arabidopsis: Type: Project View | Edit | Add Child
has Investigation, Objects: 2
EXP01: Type: Investigation View | Edit | Add Child
EXP02: Type: Investigation View | Edit | Add Child
has Material, Objects: 4
has Analysis, Objects: 3
Analysis of day 1: Type: Analysis View | Edit | Add Child
2009-09-11-+Tray1.png: Type: Data View | Edit
Analysis of day 2: Type: Analysis View | Edit | Add Child
Analysis of day 3: Type: Analysis View | Edit | Add Child
has Platform, Objects: 1
Lemnatec Scanalyser: Type: Platform View | Edit | Add Child
has Genotype, Objects: 3
Col-0: Type: Genotype View | Edit | Add Child
Gi-1: Type: Genotype View | Edit | Add Child
Gi-2: Type: Genotype View | Edit | Add Child
has Project Plan, Objects: 1
Project Plan: Type: Project Plan View | Edit | Add Child
Arabidopsis+-+University+of+Newcastle+2009-0001.xml: Type: Data View | Edit

Figure 8: The browser view of a plant phenomics project in PODD.

6. Conclusion

Sound data management practice is a challenge faced by many data-intensive scientific disciplines. A number of root causes contribute to this challenge. Firstly, scientific advancement usually connotes that the conceptual data model is being continually evolved, requiring the data management system to be sufficiently adaptive to accommodate future changes. Secondly, huge amounts of data are being generated off a wide variety of instruments and software, requiring the data management system to be highly scalable for efficient data processing. Moreover, an important requirement of data management systems is to organize data in a logical way to facilitate tasks such as curation, integration, discovery and dissemination. Hence, the management of metadata is of central importance.

Traditional data management systems are typically developed around a relational database in which database schemas define the domain model and constraints on abstract concepts. A change in schema design normally requires database migration, which is an error-prone process. As a result, such systems are not best suited in a dynamic environment where model evolution is the norm rather than the exception.

In recent years a number of repository systems such as Fedora Commons and Apache Jackrabbit have been developed. These systems impose minimal modeling constraints and hence are more flexible. As a consequence, however, rich logical structures of data cannot be easily expressed and maintained in such systems.

Given their intrinsic characteristics of semantic rigor and open nature, we believe ontology languages such as OWL and RDFS are an ideal conceptual foundation on which effective data management systems can be built. In this paper, we propose an ontology-driven architecture for developing data management systems that are able to handle dynamic data and models.

In our architecture, an ontology defines the behaviors of and relationships between domain objects using OWL vocabularies. Such definitions play a central role in all data management tasks including data acquisition, validation, presentation, discovery and integration.

We present the base ontology which encodes essential domain knowledge to describe the structure of the data through the use of OWL restrictions. The base ontology is designed in a way to strike a balance between richness in modeling capabilities and the ease of realization of data management requirements such as flexible authentication and authorization. Moreover, the base ontology can be naturally specialized to provide domain-specific definitions to cater for the different needs of individual disciplines.

Phenomics is an emergent research discipline that is poised to have a significant impact upon industrial-scale biological research. Phenomics presents a number of data management challenges, such as the management of evolving technologies and large volumes of data and the integration of highly heterogeneous datasets. Consequently phenomics represents an ideal domain for illustrating our ontology-driven approach to research data management.

To validate the feasibility of the ontology-driven architecture and to meet the data management needs of the Australian phenomics research community, we developed the PODD repository to enable efficient storage, retrieval, contextualization, query, discovery and publication of large amounts of data.

Through the employment of the ontology-driven architecture, the domain ontology and underlying repository system (Fedora Commons), the PODD repository is highly adaptive that the addition of new concepts and the modification of existing concepts do not affect data already present in the system. It is also able to perform effective data management tasks over a large and growing amount of data.

In summary, our contribution in this work is three-fold: firstly, the proposal of the ontology-driven architecture for developing data management systems; secondly, the development of a base ontology that defines essential domain knowledge; and thirdly, the development of the PODD data management system in the validation of the practicality of the proposed approach. To the best of our knowledge, this is the first proposal of an ontology-driven architecture for scientific data management systems.

We have identified a number of future work directions that we would like to pursue. Firstly, we will investigate into the integration with existing domain ontologies such as the Gene Ontology [9] and the Plant Ontology [28]. One possibility would be to use terms defined in these ontologies to annotate metadata objects [29]. Secondly, we would like to investigate the generalization of the ontology-driven approach so that it can be applied to other areas such as workflow management systems. Thirdly, we will continue the development of the PODD system to provide additional functionalities such as data visualization, automated data integration and Linked Data-style data discovery and publication.

Acknowledgement

The authors wish to acknowledge the support of the National eResearch Architecture Taskforce (NeAT) and the Integrated Biological Sciences Steering Committee (IBSSC) Australia. The authors wish to thank Dr Xavier Sirault, Dr. Kai Xu and Mr. Philip Wu for the discussion on the development of the domain ontology and the PODD system.

References

[1] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, G. Heber, Scientific data management in the coming decade, SIGMOD Rec. 34 (4) (2005) 34–41. doi:http://doi.acm.org/10.1145/1107499.1107503.

[2] A. Shah, M. Singhal, K. Klicker, E. Stephan, H. Wiley, K. Waters, Enabling high-throughput data management for systems biology: The bioinformatics resource manager, Bioinformatics 23 (7) (2007) 906–909.

[3] D. Brickley and R.V. Guha (editors), Resource Description Framework (RDF) Schema Specification 1.0, `http://www.w3.org/TR/rdf-schema/` (Feb. 2004).

[4] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language, Journal of Web Semantics 1 (1) (2003) 7–26.

[5] T. Berners-Lee, Linked Data, `http://www.w3.org/DesignIssues/LinkedData.html` (2007).

[6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007), 2008, pp. 722–735.
URL `http://dx.doi.org/10.1007/978-3-540-76298-0_52`

[7] A. Ruttenberg, J. Rees, M. Samwald, M. S. Marshall, Life Sciences on the Semantic Web: the Neurocommons and Beyond, Briefings in Bioinformatics 10 (2) (2009) 193–204. doi:10.1093/bib/bbp004.

[8] B. Smith, M. Ashburner, C. Rosse, et al., The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration, Nature Biotechnology 25 (11) (2007) 1251–1255. doi:10.1038/nbt1346.

[9] M. Ashburner, C. A. Ball, J. A. Blake, et al., Gene Ontology: Tool for the Unification of Biology, Nat Genet 25 (1) (2000) 25–29. doi:10.1038/75556. URL `http://dx.doi.org/10.1038/75556`

[10] C. Bizer, T. Heath, T. Berners-Lee, Linked data - the story so far, International Journal on Semantic Web and Information Systems (IJSWIS) 5 (3) (2009) 1–22.

[11] Y.-F. Li, G. Kennedy, F. Davies, J. Hunter, PODD: An Ontology-driven Data Repository for Collaborative Phenomics Research, in: Proceedings of 12th International Conference on Asian Digital Libraries (ICADL 2010), Springer-Verlag, 2010, to appear.

[12] D. Houle, Numbering the Hairs on Our Heads: The Shared Challenge and Promise of Phenomics, Proceedings of the National Academy of Sciences 107 (2009) 1793–1799.

[13] U. Sauer, High-throughput Phenomics: Experimental Methods for Mapping Fluxomes, Current Opinion in Biotechnology 15 (1) (2004) 58–63. doi:http://dx.doi.org/10.1016/j.copbio.2003.11.001.

[14] E. Nevo, Evolution of genome-phenome diversity under environmental stress, Proceedings of the National Academy of Sciences 98 (11) (2001) 6233–6240.

[15] R. T. Furbank, Plant phenomics: from gene to form and function, Functional Plant Biology 36 (10) (2009) 5–6.

[16] D. L. Wheeler, et al., Database resources of the national center for biotechnology information, Nucleic Acids Research 34 (Database-Issue) (2006) 173–180.

[17] A. Bairoch, et al., The Universal Protein Resource (UniProt), Nucl. Acids Res. 33 (suppl_1) (2005) D154–159. doi:10.1093/nar/gki070.

[18] C. Baru, R. Moore, A. Rajasekar, M. Wan, The SDSC storage resource broker, in: CASCON '98: Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, 1998, p. 5.

[19] W. Hoschek, F. J. Jaén-Martínez, A. Samar, H. Stockinger, K. Stockinger, Data management in an international data grid project, in: GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing, Springer-Verlag, London, UK, 2000, pp. 77–90.

[20] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, Softw. Pract. Exper. 32 (2) (2002) 135–164. doi:http://dx.doi.org/10.1002/spe.432.

[21] S. Sufi, B. Mathews, CCLRC Scientific Metadata Model: Version 2 (2004). URL `http://epubs.cclrc.ac.uk/bitstream/485/`

[22] L. N. Soldatova, R. D. King, An ontology of scientific experiments., Journal of the Royal Society, Interface 3 (11) (2006) 795–803. doi:10.1098/rsif.2006.0134.
URL `http://dx.doi.org/10.1098/rsif.2006.0134`

[23] A. Brazma, et al., Minimum information about a microarray experiment (MIAME) – toward standards for microarray data, Nat Genet 29 (4) (2001) 365–371. doi:10.1038/ng1201-365.
URL `http://dx.doi.org/10.1038/ng1201-365`

[24] A. R. Jones, M. Miller, R. Aebersold, et al., The Functional Genomics Experiment model (FuGE): an Extensible Framework for Standards in Functional Genomics, Nature Biotechnology 25 (10) (2007) 1127–1133. doi:10.1038/nbt1347.

[25] F. Manola, E. M. (editors), RDF Primer, `http://www.w3.org/TR/rdf-primer/` (Feb. 2004).

[26] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF, `http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/` (Apr. 2006).

[27] A. Rajasekar, R. Moore, F. Vernon, iRODS: A Distributed Data Management Cyberinfrastructure for Observatories, in: American Geophysical Union, Fall Meeting 2007, 2007.

[28] S. Avraham, C.-W. Tung, K. Ilic, P. Jaiswal, E. A. Kellogg, S. Mccouch, A. Pujar, L. Reiser, S. Y. Rhee, M. M. Sachs, M. Schaeffer, L. Stein, P. Stevens, L. Vincent, F. Zapata, D. Ware, The plant ontology database: a community resource for plant structure and developmental stages controlled vocabulary and annotations, Nucl. Acids Res. 36 (suppl_1) (2008) D449–454. doi:10.1093/nar/gkm908.
URL `http://dx.doi.org/10.1093/nar/gkm908`

[29] R. Schroeter, J. Hunter, A. Newman, Annotating relationships between multiple mixed-media digital objects by extending annotea, in: ESWC '07: Proceedings of the 4th European conference on The Semantic Web, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 533–548. doi:http://dx.doi.org/10.1007/978-3-540-72667-8_38.