Annalist

A practical tool for creating, managing and sharing evolving linked data

Graham Klyne *
Oxford e-Research Centre
University of Oxford
7 Keble Rd
Oxford, OX1 3QG, UK
graham.klyne@oerc.ox.ac.uk

Cerys Willoughby
Chemistry
University of Southampton
Highfield
Southampton, SO17 1BJ, UK
Cerys.Willoughby@soton.ac.uk

Kevin Page
Oxford e-Research Centre
University of Oxford
7 Keble Rd
Oxford, OX1 3QG, UK
kevin.page@oerc.ox.ac.uk

ABSTRACT

Annalist is a software system for individuals and small groups to reap the benefits of using RDF linked data, supporting them to easily create data that participates in a wider web of linked data. It presents a flexible web interface for creating, editing and browsing evolvable data, without requiring the user to be familiar with minutiae of the RDF model or syntax, or to perform any programming, HTML coding or prior configuration.

Development of Annalist was motivated by data capture and sharing concerns in a small bioinformatics research group, and customized personal information management. Requirements centre particularly on achieving low activation energy for simple tasks, flexibility to add structural details as data is collected, access-controlled sharing, and ability to connect private data with public data on the web. It is designed as a web server application, presenting an interface for defining data structure and managing data. Data is stored as text files that are amenable to access by existing software, with the intent that a range of applications may be used in concert to gather, manage and publish data.

During its development, Annalist has been used in a range of applications, which have informed decisions about its design and proven its flexibility and robustness in use. It has been particularly effective in exploring and rapid prototyping designs for linked data on the web, covering science and humanities research, creative art and personal information.

CCS Concepts

•**Information systems** → **Resource Description Framework (RDF);** *Data management systems;* •**Applied computing** → *Bioinformatics; Chemistry; Arts and humanities;*

Keywords

Semantic Web, Linked data, Data management

1. INTRODUCTION

In a blog post based on an 2013 ESWC keynote presentation, Karger[20] argues that a primary feature of Semantic Web applications should be to accommodate evolving data: "A Semantic Web application is one whose schema is expected to change". He also argues: "The current state of tools for end users to capture, communicate, and manage their information is *terrible*".

Annalist ("keeper of records") is a linked data notebook, a software system for creating, editing and managing RDF[9] linked data, which attempts to address some of the problems noted by Karger, by allowing the structure of stored data to evolve with understanding of requirements and the nature of the available data. Its primary aim is to enable individual users and small groups to reap the benefits of creating and using linked data; i.e. to create data that can be shared, evolved and re-mixed with other data on the web.

Annalist is application-agnostic, but has been developed to address data management for small research groups lacking capacity for web site development. It aims to be: easy-to-use, without programming; flexible, allowing structure to be crystallised around available data; sharable, facilitating collaboration with local and remote colleagues; and re-mixable, for combining locally created data with community resources. Annalist also "scratches an itch" as a tool for web-based personal information management and sharing.

While supporting contribution to linked data at web scale, Annalist's design assumes that individual datasets fit comfortably in the available RAM and local file system of a modern personal computer. It is not a general-purpose RDF editor, but approaches data from a perspective rooted in application concepts rather as an RDF graph, and not all RDF structures can be generated or directly managed (this does not preclude linking to arbitrary RDF data). Finally, it is not a general publishing platform: the presentation of data is oriented towards data management actives, and assume the user is familiar with the content.

Annalist is an open development[1], with source code, design notes and documentation kept in a public Github repository[2]. There is also a public demonstrator and tutorial[3].

2. MOTIVATION AND REQUIREMENTS

One motivating use case for developing Annalist was FlyTED, a database of *Drosophila* Testis Gene Expression images[39][41]. Our experiences with FlyTED (summarized in figure 1) give rise to requirements identified in parentheses, and summarized below.

*Corresponding author

WWW2016 Workshop: Linked Data on the Web (LDOW2016)

[1]http://oss-watch.ac.uk/resources/odm
[2]https://github.com/gklyne/annalist
[3]http://annalist.net/

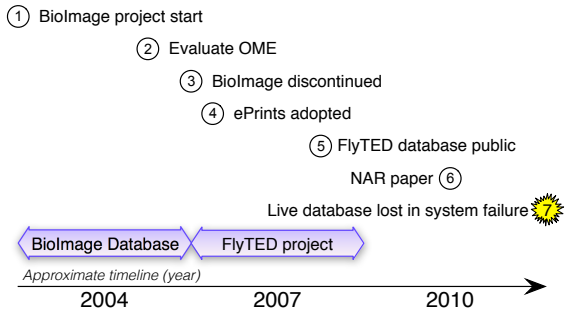

Figure 1: FlyTED database timeline.

The FlyTED database was originally intended to be published using the BioImage Database[31, 7], which was an early implementation of a database incorporating metadata based on semantic web standards. Limitations of early RDF and web software tool sets imposed design compromises that eventually led to the BioImage Database software not being sustained. Also, an early version of Open Microscopy Environment (OME)[35] was evaluated, but its metadata schema was found to be insufficiently flexible to accommodate the range of annotation information that was required (**R4**, **R5**). Eventually, a modified version of ePrints repository software[16] was used to publish the gene expression images and associated annotations. Some time after the data were published, a combination of a virtual storage service failure and a backup system configuration error resulted in the running system being lost. Although the original data remained available, the live data was stored in a relational database file; loss of the publication platform, and lack of available resources to re-apply the ePrints customizations and rebuild the database meant the database was not reinstated (**R8**, **R9**, **R10**). The loss might have been mitigated if live data had been more easily shared, including via version management systems (**R6**).

Although not large, the FlyTED data were expensive to gather, hence valuable, with each combination of gene and phenotype requiring literature and database searches, statistical analysis, laboratory procedures for sample preparation, microscopic image capture and annotation by a biological expert. Initial input of image annotations used spreadsheets, as the biologists were familiar with these (**R1**, **R3**). During the course of the investigations documented by FlyTED, the terms used to record developmental stages during which genes were active were adjusted to provide better coverage of the observations (**R5**). Programmatic access to the underlying data was subsequently used to create an exemplar application, OpenFlyData[25], which provided facilities for search and co-display of gene information across a number of *Drosophila* databases (**R11**).

The FlyTED database was created by a small team of software developers working closely with biologists, but the original intent was to pave the way for tools that biologists could use without developer support (**R1**, **R2**, **R3**). Without the support of developers, the biologists would almost certainly have not gone beyond creating spreadsheets containing their observations, the contents of which cannot easily be cross-referenced with external data sources without prior knowledge of their structure (e.g. which column is used for the FlyBase gene ID?). In creating the published database, observed genes were cross-referenced with FlyBase[12], the community database of information about *Drosophila* genes, and annotated using terms compliant with the MISFISHIE standard[10] (**R4**, **R7**).

Finally, and not directly related to the FlyTED experience, we also wanted Annalist to be suitable for offline use (e.g. for field work) (**R12**).

Summary of requirements

R1 Ease of use: possible to quickly create a simple collection and start capturing data.
R2 Ease of use: no programming or HTML coding needed to create a new collection.
R3 Ease of use: detailed knowledge of RDF and/or OWL not needed to create or edit data.
R4 Flexibility: choice of RDF vocabulary used in the data.
R5 Flexibility: possible to define or adapt structure of data as it is collected.
R6 Sharability: data can be shared between collaborators using a variety of techniques, including online access and offline file copying.
R7 Remixability: use of domain vocabularies or ontologies to facilitate combining with community datasets; ability to present data as linkable web resources, and to link to external web resources.
R8 Portability: possible to move data between live systems; not dependent on a single central service.
R9 Sustainability of software: data capture, editing and browsing possible using unmodified software.
R10 Sustainability of data: underlying data stored and exposed using a standard, easily used data format.
R11 Visible data: underlying data exposed so that functions not provided by the system (e.g. data visualization) can be implemented by independent software.
R12 Offline working: deployable on a personal computer, allowing work on linked data collections without an Internet connection.

3. RELATED WORK

As a system for creating and managing data on the web, Annalist enters a crowded space with a wealth of alternatives available. But, despite this, we are unaware of anything that provides the out-of-box capability of Annalist for creating linked data, and meeting the requirements outlined. Figure 2 gives an overview of systems with respect to the requirements listed above. Our survey focuses on tools that directly present end-user data management interfaces. There are systems (e.g. Virtuoso, Sesame, Jena, WikiBase, etc.) that are primarily semantic data stores and developer tools that are not covered. Also, tools for data cleaning (e.g. OpenRefine, LODRefine) or middleware that augments existing data (e.g. Poolparty, Sponger) are considered complementary rather than alternatives to Annalist.

3.1 Semantic Web Systems and Tools

Callimachus[4] provides flexible support for sharable linked data, but is a tool for developers rather than end-users. Semantic media wiki[5][37] is usable without additional development, but is not really suited to desktop deployment, and

[4]http://callimachusproject.org
[5]http://semantic-mediawiki.org

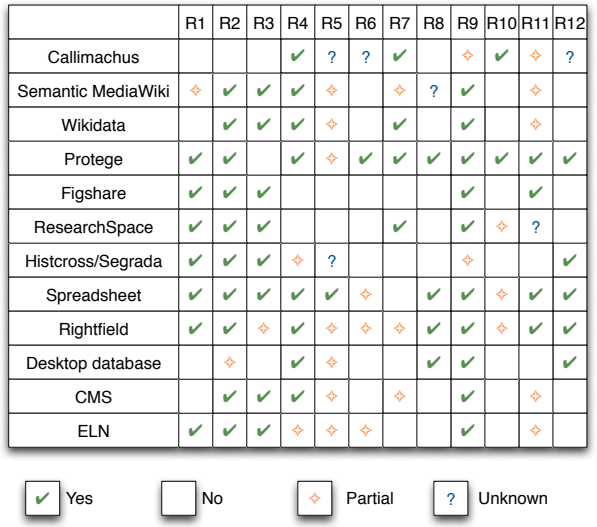

Figure 2: Related work overview.

the data is not amenable to version management or sharing via file sharing. Rauschmayer presented a poster[29] about the Hyena RDF editor at SemWiki 2008, which appears to envisage similar usage scenarios as Annalist, but does not currently appear to be available as a usable tool. This work is described in Rauschmayer's PhD thesis[28], which states that the core idea is to use a central repository for linked information, where Annalist is conceived as being just a small part in a wider linked data infrastructure. Wikidata[6], built upon the Wikibase[7] data store, acts as central storage for structured data of Wikimedia projects. It has similarities to Annalist - "items" and "statements" parallel Annalist's Entities and Fields, but the user interface is not customizable and it does not appear to support the creation of data collections independently of Wikipedia and related projects.

There are ontology design tools, such as Protege[34] (including WebProtege), which can be used to create RDF data, but a focus on ontology design leads to a complex interface that is not well suited for end-user creation and management of linked data. Changing the data structure requires an understanding of ontology design.

Piggy Bank[17] was developed as a tool for consuming web data, and creating a local RDF store to facilitate navigation and merging data from diverse sources. The emphasis here was on consuming web data from heterogeneous sources (something that Annalist can facilitate), but not so much on creating linked data for sharing and eventual publication.

Fresnel[26] is an RDF vocabulary for controlling presentation of RDF, for which Annalist uses home-spun terms. The development of Annalist focused initially on creating a user interface to create linked data without knowledge of HTML or RDF (requirements R2, R3), and the vocabulary needed to describe the presentation emerged from this approach. With the technical requirements now established in running code, evaluating a retro-fit of Fresnel could be a topic for further work. RDForms[8] ("RDF Forms") is a JavaScript library supporting a declarative description of views for editing and presenting RDF, whose interface appears to have some aspects in common with Annalist. But RDForms is a developer tool, and not something that can be used without programming.

One use for Annalist has been to create additional information, or annotations, related to web pages (e.g., see section 5.5 below). Pundit[9] is positioned as a semantic web annotation tool for research, capable of performing faceted search over annotated web pages[10]. It appears to be able to create linked data annotations, but it is not clear if it can create free-standing linked datasets, or how easily the annotations created can be exported and/or consumed by other applications. Another annotation tool is Domeo[11][8]: this, too, can create RDF annotations of online documents, but does not appear to create free-standing data.

3.2 Data sharing platforms

Figshare[12] is a a proprietary web platform for research data sharing that is well-suited for sharing research papers, supporting data and other materials, but does not of itself provide support for creating re-mixable linked data. ResearchSpace[13] is developing "a collaborative environment for humanities and cultural heritage research using knowledge representation and Semantic Web technologies", sharing some goals with Annalist, but specialized for cultural heritage by building specifically on CIDOC CRM[22].

The Database Wiki[6] is another project to provide a generic, collaborative, user-friendly interface over structured data. In this case, the underlying data is XML rather than RDF, and there is less emphasis on linking with external data. This project is informed by Form Lenses[27], a principled approach to mapping between stored data and a presented user interface, based on Applicative Functors[24], which offer a possible avenue for future work with Annalist.

Histcross[14][18] was a semantic database of historical data, subsequently replaced by Segrada[15], apparently with many similar goals to Annalist, but does not work with linked data so would not readily participate in a wider network of data.

3.3 Spreadsheets and desktop databases

Regular spreadsheets (Excel, Open Office, etc.) are very popular for research and personal information management, offering flexibility, ease of use and sharing (e.g. via CSV), but do not easily support combining data from different sources, do not provide direct web access to the underlying data, and are not particularly well suited for use with version control systems. Rightfield[38] is a tool that augments spreadsheets to facilitate entry of semantically constrained terms, and as such goes some way to addressing the remixing problems of spreadsheet data, but does not really lend itself to creating multiple cross-linked linked data structures, and shares other limitations of spreadsheets.

Bakke[1] reports an an experiment with Related Worksheets that explores the management of multiple relations between worksheets in a desktop application. Their paper

[6]https://www.wikidata.org/
[7]urlhttp://wikiba.se
[8]http://rdforms.org/
[9]http://thepund.it
[10]http://eswcdemo.gramsciproject.org
[11]http://swan.mindinformatics.org
[12]http://figshare.com
[13]http://www.researchspace.org
[14]https://github.com/mkalus/histcross
[15]http://www.segrada.org

clearly explain some problems that Annalist aims to address, and proceeds to evaluate how they can be addressed in a spreadsheet interface. The work suggests user interface designs that might be helpful, but does not of itself provide usable tools for creating linked data.

Desktop databases such as Access[16] require some configuration effort before they can be used for capturing data, which in turn are constrained by the relational schema used, and not readily linked with external datasets.

3.4 Content management systems

Other classes of web application that might be considered for research data management include Content Management Systems (CMSs, such as Wordpress[17] or Drupal[18]). These require significant development and/or configuration effort to create a data sharing platform, and do not support the full flexibility of RDF linked data. Drupal has built-in RDF support that is layered over an underlying schema, and is not amenable to change without re-working the underlying site configuration. Also, CMSs tend to hide the underlying data from direct view or manipulation, rather than exposing it for other applications to use in different ways.

3.5 Electronic Laboratory Notebooks

Annalist in some respects resembles electronic laboratory notebook systems (ELNs). There are many proprietary ELNs that are aimed at commercial research laboratories and as such may be beyond the budget of an individual or small research group. There are also some open source ELNs (e.g. Voegele et al[36], elabftw[19], LabTrove[13]. We have not specifically evaluated any of these, but in general they offer a blog-like platform, where textual notes may be augmented with named attribute or tabular data. We are not aware of any ELNs that support web linked data.

4. DESIGN

Annalist adopts a frame-oriented, or entity-oriented, approach to presenting and storing data, rather than being RDF graph based. Frames are considered to be easy for people to understand as they model some aspects of human memory patterns[18]. schraefel and Karger[30] explore in some depth the presentation of semantic web data in user interactions, and emphasize consideration of "what do we want to do"; in the case of Annalist, what we want to do is create, manage and share linked data on the web. We observe that when researchers use spreadsheets to create data, it is commonly arranged with a row of information for each of a number of similar entities (e.g. microarray descriptions commonly use a row of values for each of several thousand gene probes). The frame-based approach has implementation advantages, too: it provides a convenient grouping of data such that the description of each entity is stored as a separate file, assigned a URL, and directly accessed as a web resource.

In discussions with researchers about their preference for using spreadsheets, we were told that one of their reasons is that spreadsheets do not impose *a priori* constraints on what can be entered, making it easy for them to enter data as it becomes available. Bakke *et al*[1] also note "When it comes to general editing tasks on tabular data, spreadsheet systems have an advantage even over most tailor-made applications", and advantages of a system that can "allow temporary inconsistencies". Frey *et al*[13] also note that semantically aware tools "could be too heavyweight and prescriptive", restricting re-use in other areas. Annalist adopts a principle that its first task is to make it easy for researchers to capture their data; defining structure is secondary. Further, there is no attempt to validate data entered, or impose any kind of quality standards: we take a view that validity, quality and refinement are dependent on a context of use[40], and as such are usefully applied in such context.

Linked data vocabulary terms are commonly associated with schemas (or ontologies), but we observe that such terms may be adopted independently of any schema. Annalist supports evolving data initially though addition of terms, even to the extent of taking an unstructured narrative, identifying significant elements, and progressively articulating them using new terms, preferably drawn from existing stable schemas (e.g. sections 5.1 and 5.2). A related concern is evolving schemas, which incur changes to terms already used in data. Annalist provides support for supertype URIs in type definitions, and property aliases, which can assist with type and property URI migration; generalization of these features is ongoing. We have also performed migrations by direct editing of the underlying data; while not an option for non-technical users, it shows that schema evolution can also be assisted by external services.

The requirements for internal data to be exposed to third party applications, and data portability, are addressed by using JSON – specifically JSON-LD[33] – as the primary internal data storage format. JSON-LD conventions allow data to be interpreted as RDF, yet retaining the ease-of-use of JSON, and use by applications that have no knowledge of RDF. Within the data managed by Annalist, internal links are stored as URI relative references[4], to be resolved against the URL used to access the data, allowing data to be copied from one Annalist deployment to another without changes. Use of Compact URIs (CURIEs[20]) for field names and types, with prefixes defined in external JSON-LD context files, allows for more compact, readable and understandable data compared with using full URIs.

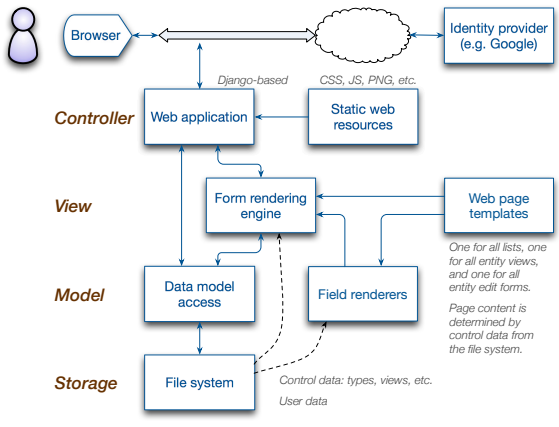

Figure 3: Annalist software components.

[16]https://products.office.com/en-us/access
[17]https://wordpress.org
[18]https://www.drupal.org
[19]http://elabftw.net/
[20]http://www.w3.org/TR/curie/

Figure 3 shows the main components of the Annalist software. It is implemented as an HTTP server application, written in Python using the Django[21] software framework. All the essential Annalist application logic is implemented by the server, but there is some limited use of Javascript in the browser to provide a more responsive user interface; the intent here is that Annalist can be used in browser enviroments where Javascript is disabled or unavailable. The Annalist server software is designed to be deployed locally (on a personal computer), on a private network, or on a publicly accessible host.

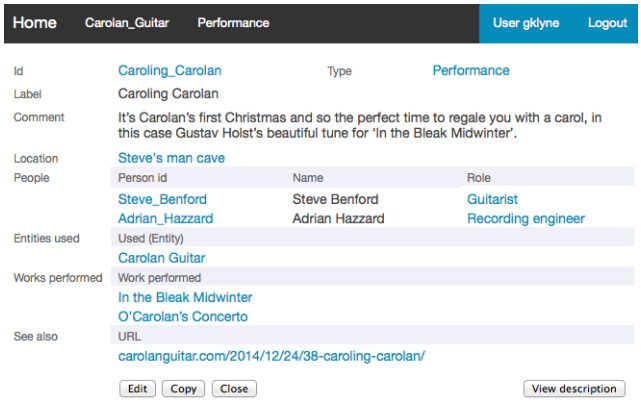

Figure 4: Annalist data record view example.

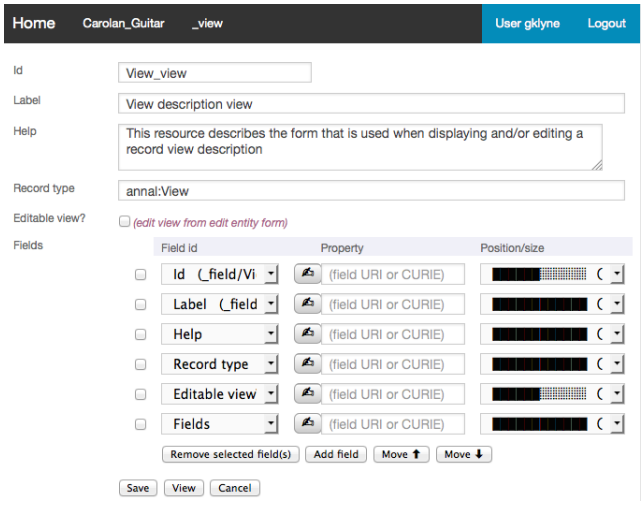

Figure 5: Annalist record edit view example.

At the heart of Annalist is a dynamic web-page creator and form rendering engine that combines user data with a form description to create an HTML web page. Figure 4 is an example of data viewed using Annalist. The underlying JSON-LD data can be accessed by web retrieval, either via Annalist, or a suitably configured web server; the file layout is designed to preserve relative references. This helps to ensure that access to the data is not dependent on the health of the Annalist service. A goal is to allow user data to be stored on the web, separately from the Annalist service itself, though

[21]https://www.djangoproject.com

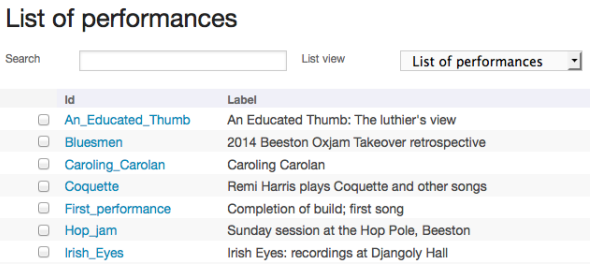

Figure 6: Annalist list view example.

there remain some access control details to be resolved to make this a reality.

Figure 5 shows an example of the Annalist data editing interface. It actually shows a form used for editing the definition of a form description, so is self-referential: the labels of the fields on the left of the page are echoed in the list of field descriptions at the bottom of the page.

Figure 6 shows an excerpt from a listing of records. These examples illustrate main kinds of display provided by Annalist: a detailed view of a single record, which may be an editing view or a view-only display, and a summary list of multiple records. Further examples of the Annalist user interface can be found in the Annalist tutorial document[22].

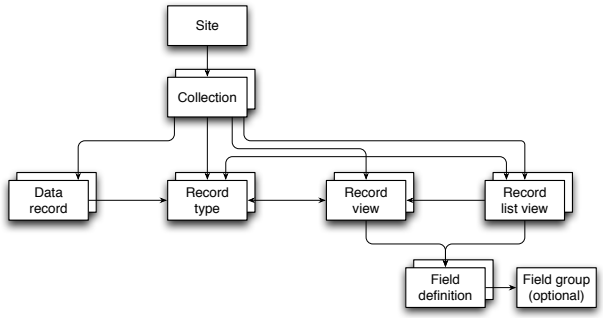

Figure 7: Annalist internal data model.

Like the user data, form descriptions and other configuration data are stored as JSON-LD files, and are editable through the Annalist web interface. This makes Annalist self-maintaining, in the sense that there is no separate configuration interface or other mechanism needed to define data types, storage structures, or their presentation.

Data are organized as illustrated in figure 7. At the top level is an Annalist ***Site***, which is associated with an Internet host (or `localhost` for desktop deployment). Site data is grouped into free-standing ***Collections***, which contain user data, and metadata to define its structure and presentation.

The data are stored in ***Data records***, each of which is presumed to describe some entity, and corresponds to an addressable web resource or file (each having a distinct URL). ***Record types*** correspond to the type of entity described, and are used to combine similar entities for user presentation (e.g. in lists), and also in the underlying data storage (e.g. entities of different types are stored in different directories or storage containers). ***Record views*** define forms used

[22]http://annalist.net/documents/tutorial/main

for creating, editing or viewing data records; ***Record lists*** define presentation of multiple entities for browsing. ***Field definitions*** are referenced by *record views* and *record lists*, and control the internal representation and presentation of record component values. ***Field groups*** are used to group fields for various purposes, e.g. to define repeated groups of fields. URIs used for type and property URIs are contained in the record type, field and view definitions, and may be drawn from existing ontologies, or local ad/hoc identifiers with potential to adopt existing vocabularies as correspondences are determined.

Access control is managed in two parts: authentication is by a third-party identity provider (IDP) using OpenID Connect[23] returning an authenticated email address. Annalist has been tested to date using the Google login service[24]. Access control is handled by permissions stored as Annalist records, which are defined and applied on a per-collection basis, with fall-back to site-wide permissions. Permissions required for access may depend on record type (e.g. ADMIN permission required to access the permission records), and in future this might be used for finer-grained control.

5. APPLICATIONS

Annalist has been used with several personal and research projects, described below, which have informed its ongoing development. The first example includes a sketch of its implementation to provide insight into use of Annalist, and all can be examined at the URLs given.

5.1 The Carolan Guitar

This is a project of Nottingham University's Mixed Reality Laboratory on "Augmenting a Guitar with its Digital Footprint"[3], recording its history online in the form of a blog[25]. Annalist has been used to create a linked data overlay of this history that links to the blog itself, and also to other key resources that are part of its history[26]. This overlay models events (construction, composition, performance and others), people, places, artifacts, materials, musical compositions and more using vocabularies drawn from RDFS[5], CIDOC CRM[22, 11], FRBRoo[2], and W3C PROV[23].

5.1.1 Carolan Guitar implementation

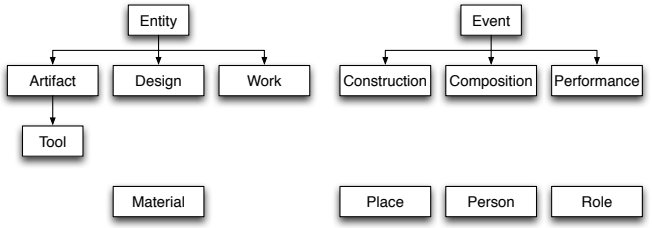

Figure 8: Carolan Guitar description types.

The Carolan Guitar description is built around the types shown in figure 8. The types `Entity` and `Event` reflect core elements of both PROV (`prov:Entity`, `prov:Activity`) and CIDOC CRM (`crm:E71_Man-Made_Thing`, `crm:E5_Event`) ontologies. Other, more refined, types are introduced as judged useful to capture the guitar's history. The Carolan Guitar itself is an instance of `Artifact`, a subtype of `Entity`, which is primarily an instance of the CIDOC CRM type `crm:E24_Physical_Man-Made_Thing`, but is also associated with a number of other types in the type definition[27]:

```
{"annal:id":    "Artifact",
 "annal:type":  "annal:Type",
 "rdfs:label":  "A constructed physical entity",
 "rdfs:comment": "An artifact, such as a musical
     instrument or some other object.",
 "annal:uri":   "crm:E24_Physical_Man-Made_Thing",
 "annal:supertype_uris": [
   {"annal:supertype_uri": "prov:Entity"},
   {"annal:supertype_uri": "crm:E77_Persistent_Item"},
   {"annal:supertype_uri": "crm:E70_Thing"},
   {"annal:supertype_uri": "crm:E71_Man-Made_Thing"},
   {"annal:supertype_uri": "crm:E18_Physical_Thing"},
   {"annal:supertype_uri": "frbroo:F7_Object"}],
 "annal:type_list": "_list/Artifacts",
 "annal:type_view": "_view/Artifact"}
```

The central subject, the Carolan Guitar, is presented using the `Artifact` Record view[28]. This view describes an `Artifact` with an identifier, type, label, description, links to further information, and (central to this application) a list of life events, which correspond to a journal of its history.

Information about the Carolan Guitar and its life events are recorded in its description[29], e.g.:

```
  :
 {"crm:P12i_was_present_at":
     "Construction/Construction_9"},
 {"crm:P12i_was_present_at":
     "Performance/First_performance"},
 {"crm:P12i_was_present_at": "Performance/Stairway"},
 {"crm:P12i_was_present_at": "Performance/Hop_jam"},
 {"crm:P12i_was_present_at": "Event/Photo_shoot"},
 {"crm:P12i_was_present_at":
     "Composition/Catch_the_moment"},
  :
```

Here, relative URL references are used to designate life events, each of which may record information about where it took place, entities used, and who was involved in what roles. Different types of event in the guitar's history (construction, composition, performance, etc.) may also have different information: a construction event view[30] may include information about the tools, materials used; a performance view[31] may include details of the works performed.

The modeling of the Carolan Guitar's story is by no means complete (if such a thing is ever possible), and some choices of what to include could reasonably be described as arbitrary. But, using Annalist, the description can be augmented and

[23] http://openid.net/connect/
[24] https://developers.google.com/identity/protocols/OpenIDConnect
[25] http://carolanguitar.com
[26] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/Artifact/Carolan_Guitar/
[27] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/_type/Artifact/type_meta.jsonld
[28] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/_view/Artifact/view_meta.jsonld
[29] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/Artifact/Carolan_Guitar/entity_data.jsonld
[30] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/_view/Construction/view_meta.jsonld
[31] http://demo.annalist.net/annalist/c/Carolan_Guitar/d/_view/Performance/view_meta.jsonld

refined with additional types and more detailed view descriptions as new requirements are encountered. Indeed, this has already happened several times during its development.

5.2 Smoke: creating an audio-visual poem

Procedural Bending is presented in Garrelfs' PhD thesis[14] as a model for discourse about creative processes. It has similarities with the W3C PROV model[23], but also some key differences. Annalist has been used to create a description of the creation of "Smoke"[32], an "experimental documentary come audio-visual poem" about mid 20th-century air pollution in the cities of Pittsburgh and St Louis. In this case, a semi-structured blog-like journal was created by the artist using Annalist[33], together with a Procedural Blend diagram using the model from Garrelfs' thesis. We worked with the artist to encode the blend diagram as Annalist records, and in the process were able to refine the model to make it more consistently encodable while preserving the original descriptive intent.

5.3 Chemistry Personas

Chemistry Personas[34] evaluates Annalist as a tool for capturing records of academic researchers in chemistry, and for identifying metadata from these records. It was used to create a set of interfaces for the capture and linking of information about people, organisations, projects, and resources associated with experiments such as plans, materials, equipment, activities, and the experiment records themselves that incorporate linked data. Designs of the models are based on research information and associated metadata from experiment records, and observed recording practices in chemistry from a range of universities across the world.

Annalist was found to be useful for capturing research records and associated data, with the flexibility to be easily adapted to the needs of different research groups and individual researchers. We found the generic capability to create specialized interfaces for capturing information allowed us to handle the different requirements of different domains and disciplines. The linked-data aspect is particularly useful in enabling the simple reuse of resources and plans, and inclusion of frequently used information into the research records.

5.4 Canal Cruising Log

The canal cruising log[35] is an example of Annalist used for personal information management. It captures information about narrowboat cruising on the English canal network, with information about daily movements, waterways, places visited, other interesting locations, and maintenance activities performed. It is based on a handwritten log book, and attempts to capture information in searchable form that may be useful when planning waterways travels. The information modelling is *ad hoc* (i.e. uses private vocabulary terms). Using Annalist, it would be quick and easy to revisit and add class URIs from standard ontologies. Property URIs are harder to update, but but work is in progress to support data migration as properties change.

[32]http://irisgarrelfs.com/smoke

[33]http://cream.annalist.net/annalist/c/IG_Philadelphia_Project/

[34]http://cream.annalist.net/annalist/c/Chemistry_Personas/

[35]http://demo.annalist.net/annalist/c/CruisingLog/

5.5 Accommodation search

The accommodation search collection[36] is another example of personal information management, this time with a clear real-world outcome. We sought a new home for an elderly relative that would make it easier to provide increasingly-needed levels of support. The web made it easy to find candidate properties, but there were specific requirements (e.g., physical accessibility) not selectable by available search facilities, so we had to filter from a large number of candidate properties; further, good properties would come to market and disappear quickly, so prompt information sharing was needed. Annalist was used to rapidly create a specialised database of candidates, with links to existing property web sites, additional annotations, an an overall scoring of suitability according to our particular requirements. This was shared among family members, and when the ideal property appeared we were able to consult over the details and arrange an early visit.

6. DISCUSSION

6.1 Evaluation of requirements

Section 2 set out a number of requirements arising mainly from past experiences creating FlyTED. We now review those requirements against the implemented Annalist applications described in section 5.

Requirements **R2** (no programming), **R6** (data sharing), **R9** (use of unmodified software) and **R10** (expose data in a standard format) are demonstrated by all of the implementations described. Our work on these implementations has repeatedly exploited **R8** (portability of data) and **R12** (offline working) by using a GitHub repository[37] to transfer work-in-progress data between an offline laptop and online servers; this use of Github for data exchange, backup and versioning also demonstrates another aspect of **R10** (sustainability of data). Annalist's exposure of underlying data (**R11**) is present in all the applications described, but has not been significantly exploited by independent software; this will be tested in future work.

The implementation of *The Carolan Guitar* data has shown extensive use of **R4** and **R7** (choice and mixing of existing RDF vocabularies), combining PROV, CIDOC CRM, FRBRoo and some private terms. Requirement **R5** (evolution of structure) was used when working through the Carolan Guitar's history, starting with its construction, but in later stages dealing with musical performances and compositions.

The *The Canal Cruising Log* implementation in particular demonstrates **R3** (not needing knowledge of RDF) as this has been developed without reference to any specific RDF terms or characteristics. The frame-oriented approach to data presentation appears to be more approachable than requiring users to work with the RDF graph/triple structure (also suggested by Kalus[18]). There is one aspect where RDF influences remain visible: field descriptions must specify a property string in the form of a URI or CURIE that is used to identify an attribute in the data (the effect of which is to constrain the syntax of attribute identifying strings).

The *Accommodation search* application demonstrates the achievement of **R1** (to quickly create a collection and start

[36]http://demo.annalist.net/annalist/c/Accommodation_search/

[37]https://github.com/ninebynine

capturing data) as this ephemeral application would never have been realized if it would have taken more than an hour or so to create the collection with some initial data records.

6.2 Further observations

Primary storage of data as simple text resources: there is no database or triple store behind Annalist. For locally stored resources, each addressable resource is stored as a single JSON-LD file, and the request URL is mapped to a corresponding filename. The underlying resources may be served directly by a web server without any Annalist deployment being present, which we believe could be a boon for long-term sustainability of research data (e.g., the Annalist demo site[38] also offers links[39] that connect directly to the underlying data). This approach allows collections to be versioned using common version management tools (e.g. git), and shared via version repositories.

Edit conflicts: Annalist handles concurrent updates to individual entities by atomic updates. There is an unimplemented design for warning when an entity changes during editing. Consistency of values between entities is not currently enforced (see section 4).

Performance: the Annalist demonstrator runs on a modestly provisioned virtual machine (1 virtual CPU, 2Gb RAM). We have not undertaken formal measurements, but have not found performance to be a limitation in day-to-day use. This matches our expectation that the underlying Linux file system is very efficient for accessing small files that comprise Annalist collections. Some particular operations perform repeated file accesses, and future work is planned to optimize these cases. Planned developments will introduce an index alongside the flat files, probably a triple store, to support efficient search and query over the data.

Data types for organisation, views to define structure: traditional database systems use data types (or equivalent) both to categorize data records and to associate structure with the data (e.g. a relational table defines the structure of each row in a table). This is less true for schema-free databases (e.g. MongoDB[40]), but even here we may see that structural features such as indexes are associated with what is effectively a data type (e.g. in MongoDB, a "collection"). With Annalist, types are used simply to categorize data records, and the structure of any record is determined by the view (or views) that are used to create or edit it. This means that different views can be used with a record type, according to the context (e.g., considering a person as an employee or as a customer). (When editing a record with fields not referenced by the view used, those fields are unchanged when the record is saved.)

Collection portability: URLs and URIs: moving a collection between Annalist deployments means that the URLs used to access records can change. But for some fields, such as type descriptions, we need stable identifiers that don't change with location of the data. The implementation of Annalist recognizes this tension, and distinguishes between URLs used to access and retrieve resources and URIs used to identify them. This goes somewhat against the grain of web wisdom, and in practice the distinction is used quite sparingly, but we believe this illustrates that in pragmatic applications, particularly where there may be copies of the same information in different locations, the distinction may have some value. (This differs from earlier discussions about URNs and URIs[41], in that the distinction is in no way dependent on the URI scheme used: a given HTTP URI may be a URI or URL depending on how it is used.)

Usability: we have not yet undertaken formal usability studies, but our experience to date indicates that it is possible to quickly create data managment forms that are usable with no special knowledge or experience. We have found that creating structure and view definitions can become complicated when there are multiple relationships between entity types, and improvements in this area are ongoing.

Scale: Annalist deals with collections of modest size, but through the milieu of the web even such modest collections, in sufficient numbers, may contribute to data at much greater scales. Tools, like Annalist, that facilitate creation of linked data at local scales may be a key to enabling fully distributed datasets at web scale.

7. FURTHER WORK

The nature of Annalist as a generic tool means there is inevitably far more that could be done than has been achieved to date. Work-in-progress enhancements include: modular type/view definitions, importing definitions from a predefined collection, which we see as allowing end-users to get started even more quickly with generating their own linked data (e.g. using "canned" definitions for bibliographic and provenance information); usability improvements to streamline common data entry tasks (e.g. automatic creation of default views and lists associated with a data type); data migration facilities to assist with applying vocabulary changes to existing data.

Work is currently underway to create an independent front-end for presentation of musical performance data created using Annalist (based on the structures developed for the Carolan Guitar), which we plan to use to develop a demonstration system aimed at enhancing audience experience of live music concerts. This will provide an exemplar of how Annalist may be used as part of a larger ensemble of tools for creating and deploying applications using linked data.

We have noted that Annalist is not a "big data" system, and that design choices may constrain the effective size of a single Annalist collection. But by creating multiple independent, cross-linked and web-searchable data, we anticipate that Annalist collections may be combined with other data sources, contributing to creation of linked data resources at larger scale. One way to explore this idea would be to use Annalist to reinstate public access to the FlyTED data. Some initial explorations are under way, and successful achievement of this could provide a particularly compelling evaluation of the Annalist principles and design.

Looking ahead, we anticipate creation of "data bridges" to allow existing data (especially in spreadsheets) to be presented as linked data through Annalist; this might extend further to real-time data acquisition, with data from sensors like GPS or real-time feeds.

The current implementation of Annalist uses the server host file system for data storage, but it was an original goal that Annalist could work with third-party storage. A candidate for this would be a Linked Data Platform (LDP)[32] server, though there are matters of access control to be

[38] http://demo.annalist.net/
[39] http://annalist.net/annalist_sitedata/
[40] https://www.mongodb.com/
[41] http://www.w3.org/TR/uri-clarification/

resolved. The Social Linked Data (Solid) project[42] uses WebId[43] for access control, and could provide an promising opportunity if it is possible to devise a mechanism to link OpenID Connect authentication with WebId authorization.

Other areas of possible future work for Annalist include provenance[15] recording and support for provenance pingbacks[21] to recognize downstream use of Annalist data.

Looking further ahead, we note that the Annalist dynamic form generator has evolved in a somewhat ad hoc fashion, in response to evolving recognition of requirements. As noted in section 3.2, theoretical work on "form lenses"[27] might be adopted to provide a more principled grounding for this aspect of Annalist.

Finally, we note that while Annalist has been an open development from its outset, it has been significantly conducted so far by an "open community" of just one developer. For a viable future we must engage a wider community of users and developers to establish a more long-term sustainable project. Contributions will be most welcome!

8. CONCLUSIONS

Annalist has been used in a number of research projects for prototyping linked data information structures and, even as a work-in-progress, has proven flexible and robust in use by a small number of diverse users, with a low cost to get started with a new collection. It has also proved effective in personal information management projects involving annotation of existing web resources and sharing structured data on the Web. While many of the capabilities of Annalist are provided by other systems, we are not aware of any other that combines the key features of Annalist in a package that can be used "out of the box" for data management.

We have noticed that, while Annalist is easy to use for basic data entry and browsing, developing more complex structures requires greater effort, and does benefit from an awareness of the RDF model (particularly with respect to the use of URIs, or CURIEs, to identify things, classes of things and relations between things). But this effort can be applied incrementally, yielding rapid benefits and feedback, supporting agility in creating information designs.

Annalist's flexible approach to information structuring has permitted an approach that differs from that often used when creating databases (e.g. for research data), starting with a very loosely structured narrative and progressively refining structured information around that narrative. In this, we feel that we have created a tool that goes some way to achieving Karger's objectives for semantic web applications, *viz.* "to work effectively over whatever schemas their users choose to create or import"[19].

9. ACKNOWLEDGEMENTS

The development and evaluation of Annalist has been supported in part by EPSRC EP/L019981/1 Fusing Semantic and Audio Technologies for Intelligent Music Production and Consumption and by the JISC-funded Research Data Spring CREAM project[44].

[42]https://github.com/solid/solid-spec

[43]http://www.w3.org/2005/Incubator/webid/spec/identity/

[44]https://blog.soton.ac.uk/cream/

References

[1] E. Bakke, D. Karger, and R. Miller. A spreadsheet-based user interface for managing plural relationships in structured data. In *Proc. SIGCHI conference on human factors in computing systems*. In CHI '11. ACM, Vancouver, 2011, pp. 2541–2550. DOI: 10.1145/1978942.1979313.

[2] C. Bekiari, M. Doerr, and P. Le Bœuf, eds. *FRBR - object-oriented definition and mapping to FRBRer (version 1.0)*. 1.0 ed. May 2009, 1.0 ed., 2009. URL: http://cidoc.ics.forth.gr/docs/frbr_oo/frbr_docs/FRBRoo_V1.0_draft__2009_may_.pdf.

[3] S. Benford, A. Hazzard, et al. Augmenting a guitar with its digital footprint. In *Proc. international conference on new interfaces for musical expression*. Louisiana State University, 2015, pp. 303–306. URL: http://www.nime.org/proceedings/2015/nime2015_264.pdf.

[4] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. 2005. URL: https://tools.ietf.org/html/rfc3986.

[5] D. Brickley and R. Guha. RDF schema 1.1. W3C Recommendation. World Wide Web Consortium, Feb. 2014. URL: http://www.w3.org/TR/rdf-schema/.

[6] P. Buneman, J. Cheney, et al. The Database Wiki project: a general-purpose platform for data curation and collaboration. *SIGMOD record*, 40(3):15–20, 2011. DOI: 10.1145/2070736.2070740.

[7] C. Catton, S. Sparks, and D. M. Shotton. Chapter 21: publishing and finding images in the BioImage Database, an image database for biologists. In, *Cell biology (third edition)*, pp. 207–216. Academic Press, Burlington, third edition ed., 2006. DOI: 10.1016/B978-012164730-8/50149-0.

[8] P. Ciccarese, M. Ocana, and T. Clark. Open semantic annotation of scientific publications using DOMEO. *J. biomedical semantics*, 3(S-1):S1, 2012. URL: http://www.jbiomedsem.com/content/3/S1/S1.

[9] R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 concepts and abstract syntax. W3C Recommendation. Feb. 2014. URL: http://www.w3.org/TR/rdf11-concepts/.

[10] E. W. Deutsch et al. Development of the minimum information specification for in situ hybridization and immunohistochemistry experiments (MISFISHIE). *Omics: a journal of integrative biology*, 10(2):205–208, 2006.

[11] M. Doerr. The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI mag.*, 24(3):75–92, Sept. 2003. URL: http://dl.acm.org/citation.cfm?id=958671.958678.

[12] G. dos Santos et al. Flybase: introduction of the *Drosophila Melanogaster* release 6 reference genome assembly and large-scale migration of genome annotations. *Nucleic acids research*, 43(Database-Issue):690–697, 2015. DOI: 10.1093/nar/gku1099.

[13] J. G. Frey, A. J. Milsted, et al. Labtrove: a lightweight, web based, laboratory blog as a route towards a marked up record of work in a bioscience research laboratory. *Plos one*, 8(7):e67460–[18pp], July 2013. DOI: 10.1371/journal.pone.0067460. URL: http://eprints.soton.ac.uk/355078/.

[14] I. Garrelfs. From inputs to outputs: an investigation of process in sound art practice. PhD thesis. University of the Arts London, May 2015. URL: http://irisgarrelfs.com/thesis.

[15] P. Groth and L. Moreau. PROV overview: an overview of the PROV family of documents. W3C Working Group Note. W3C, Apr. 2013. URL: http://www.w3.org/TR/prov-overview/.

[16] C. Gutteridge. GNU EPrints 2 overview. In *11th panhellenic academic libraries conference*. Event Dates: 2002, 2002. URL: http://eprints.soton.ac.uk/256840/.

[17] D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: experience the semantic web inside your web browser. *Web semantics: science, services and agents on the World Wide Web*, 5(1), 2007. DOI: 10.1016/j.websem.2006.12.002.

[18] M. Kalus. Semantic networks and historical knowledge management: introducing new methods of computer-based research. *Journal of the Association for History and Computing*, 10(3), Dec. 2007. URL: http://hdl.handle.net/2027/spo.3310410.0010.301.

[19] D. Karger. Keynote at ESWC part 2: how the semantic web can help end users. Tech. rep. MIT CSAIL Research, 2013. URL: http://haystack.csail.mit.edu/blog/2013/06/06/keynote-at-eswc-part-2-how-the-semantic-web-can-help-end-users/.

[20] D. Karger. Keynote at the ESWC part 1: the state of end user information management. Tech. rep. MIT CSAIL Research, 2013. URL: http://haystack.csail.mit.edu/blog/2013/06/05/keynote-at-the-european-semantic-web-conference-part-1-the-state-of-end-user-information-management/.

[21] G. Klyne and P. Groth. PROV-AQ: provenance access and query. W3C Working Group Note. W3C, Apr. 2013. URL: http://www.w3.org/TR/prov-aq/.

[22] P. Le Bœuf, M. Doerr, et al. Definition of the CIDOC conceptual reference model, version V6.2. Tech. rep. International Council of Museums, May 2015. URL: http://www.cidoc-crm.org/docs/cidoc_crm_version_6.2.pdf.

[23] T. Lebo et al. PROV-O: the PROV ontology. W3C Recommendation. W3C, Apr. 2013. URL: http://www.w3.org/TR/prov-o/.

[24] C. McBride and R. Paterson. Applicative programming with effects. *J. funct. program.*, 18(1):1–13, Jan. 2008. DOI: 10.1017/S0956796807006326.

[25] A. Miles, D. Shotton, et al. OpenFlyData: an exemplar data web integrating gene expression data on the fruit fly *Drosophila Melanogaster*. *Journal of biomedical informatics*, 43(5):752 –761, 2010. DOI: http://dx.doi.org/10.1016/j.jbi.2010.04.00.

[26] E. Pietriga, D. Karger, et al. Fresnel: a browser independent presentation vocabulary for RDF. In *Proc. 5th international conference on the semantic web*. In ISWC'06. Springer-Verlag, Athens, 2006, pp. 158–171. DOI: 10.1007/11926078_12.

[27] R. Rajkumar, S. Lindley, et al. Lenses for web data. *Electronic communications of the EASST*, 57(57), 2013. DOI: 10.14279/tuj.eceasst.57.879.

[28] A. Rauschmayer. Connected information management. PhD thesis. Ludwig-Maximilians-Universität München, Feb. 2010. URL: http://nbn-resolving.de/urn:nbn:de:bvb:19-114390.

[29] A. Rauschmayer. Hyena RDF editor. Poster. Displayed at SemWiki 2008, 3rd Semantic Wiki Workshop: The Wiki Way of Semantics. 2008. URL: http://ceur-ws.org/Vol-360/poster-11.pdf.

[30] m. schraefel and D. Karger. The pathetic fallacy of RDF. In *International workshop on the semantic web and user interaction (SWUI) 2006*. URL: http://eprints.soton.ac.uk/262911/.

[31] D. M. Shotton, S Sparks, and C Catton. An introduction to the BioImage database, an image database for biologists. In *Proc. 4th european light microscopy initiative meeting, Gothenburg*, 2004.

[32] S. Speicher, J. Arwe, and A. Malhotra. Linked data platform 1.0. W3C Recommendation. Feb. 2015. URL: http://www.w3.org/TR/ldp/.

[33] M. Sporny, D. Longley, et al. JSON-LD 1.0: a JSON-based serialization for linked data. W3C Recommendation. Jan. 2014. URL: http://www.w3.org/TR/json-ld/.

[34] Stanford Center for Biomedical Informatics Research (BMIR). Protégé ontology editor. URL: http://protege.stanford.edu.

[35] J. R. Swedlow et al. The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging. *Genome biology*, 6(5):R47–R47, 2005. DOI: 10.1186/gb-2005-6-5-r47.

[36] C. Voegele et al. A universal open-source electronic laboratory notebook. *Bioinformatics*, 29(13):1710–1712, 2013. DOI: 10.1093/bioinformatics/btt253.

[37] M. Völkel and M. a. Krötzsch. Semantic wikipedia. In *Proc. 15th international conference on World Wide Web*. ACM, Edinburgh, Scotland, 2006, pp. 585–594. DOI: 10.1145/1135777.1135863.

[38] K. Wolstencroft et al. RightField: semantic enrichment of systems biology data using spreadsheets. In *8th IEEE international conference on e-science, e-science 2012, Chicago, IL, USA,* October 8-12, 2012, pp. 1–8. DOI: 10.1109/eScience.2012.6404412.

[39] J. Zhao, G. Klyne, E. Benson, E. Gudmannsdottir, H. White-Cooper, and D. Shotton. FlyTED: the *Drosophila* testis gene expression database. *Nucleic acids research*, 38(Suppl. 1):D710–D715, 2010. ISSN: 0305-1048. DOI: 10.1093/nar/gkp1006.

[40] J. Zhao, G. Klyne, M. Gamble, and C. Goble. A checklist-based approach for quality assessment of scientific information. In *3rd international workshop on linked science (LISC2013)*, 2013. URL: http://linkedscience.org/wp-content/uploads/2013/04/paper5.pdf.

[41] J. Zhao, G. Klyne, and D. Shotton. Building a semantic web image repository for biological research images. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *The semantic web: research and applications*. Vol. 5021, in Lecture Notes in Computer Science, pp. 154–169. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-68234-9_14.