Annalist

("keeper of records")

Graham Klyne
e-Research Centre, University of Oxford

http://annalist.net

Oxford e-Research Centre

Digital Economy
Transforming Business and Society

Queen Mary University of London

EPSRC
Engineering and Physical Sciences Research Council

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Acknowledgements

OeRC: FAST project (digital music)

- Kevin Page, David Weigl, David De Roure, Terhi Nurmikko-Fuller, Alan Chamberlain, Steve Benford, *et al*

JISC RDS: CREAM project (active metadata)

- Cerys Willoughby, Simon Coles, Colin Bird, Iris Garrelfs, Athanasios Velios, Mike Mineter, *et al*

Wf4Ever (research objects, workflows)

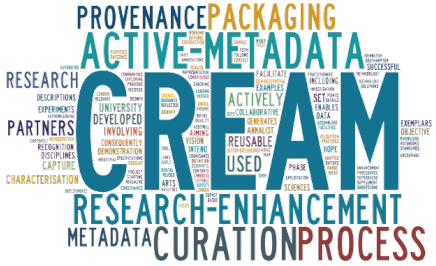
- Jun Zhao, Kevin Page, David De Roure, Stian Soiland-Reyes, Sean Bechhofer, Khalid Belhajjame, Carole Goble, Daniel Garijo, Oscar Corcho, *et al*

OeRC: Claros project (classical art)

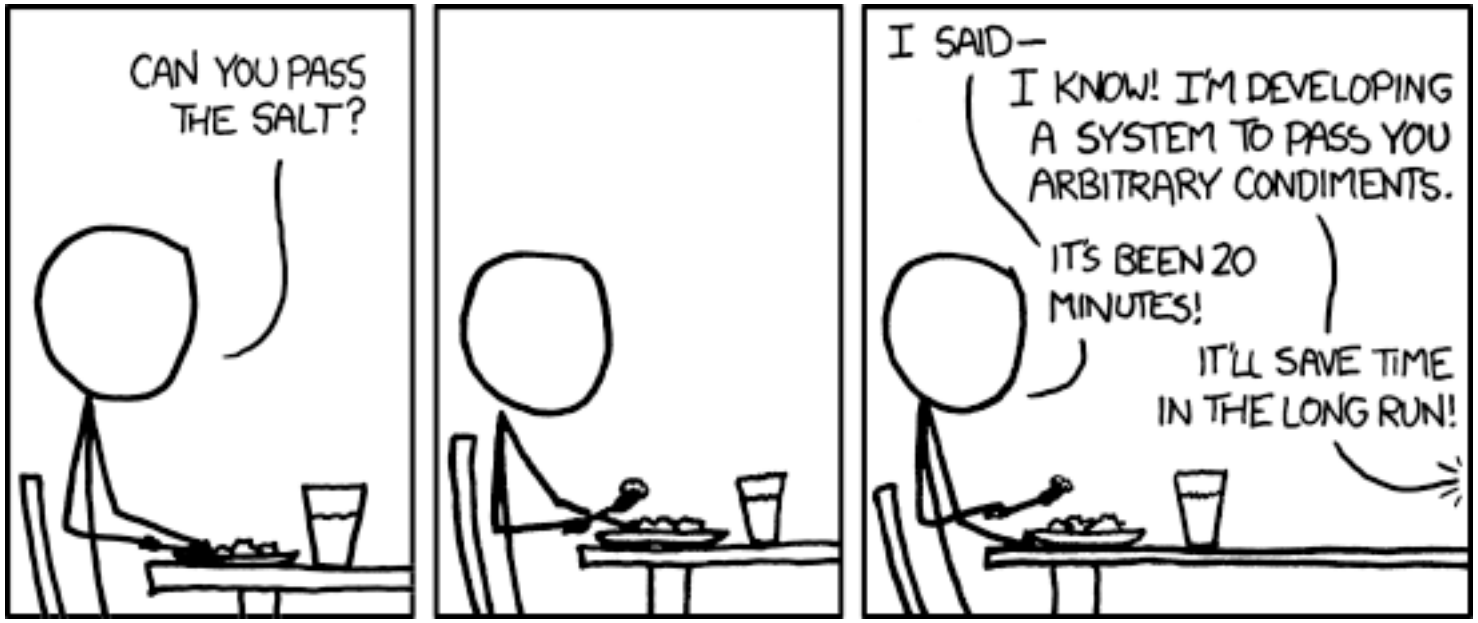
- Donna Kurtz, Robert Kummer, Reinhard Förtsch, *et al*

Oxford (Zoology): IBRG (image bioinformatics)

- David Shotton, Jun Zhao, Alistair Miles, Helen White-Cooper, *et al*



Origins



https://xkcd.com/974/

Goal

To make it quick and easy for individuals and small teams to create and share linked data on the web

Motivation and Requirements

Annalist in use

Design and implementation

Status and future work

Discussion

Example: Fly-TED

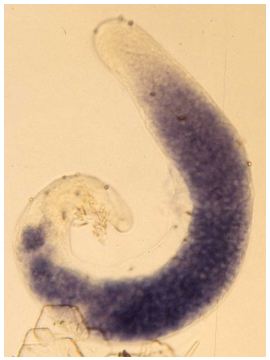
These *in situ* hybridization images show gene expression at different stages of spermatogenesis, created by a complex laboratory process.

Each image corresponds to a different combination of gene and a strain of *Drosophila melanogaster* (fruit fly).

Interpretation and reproducibility require preparatory steps to be recorded along with the images and annotations.

CG2247 wt

CG2247 topi

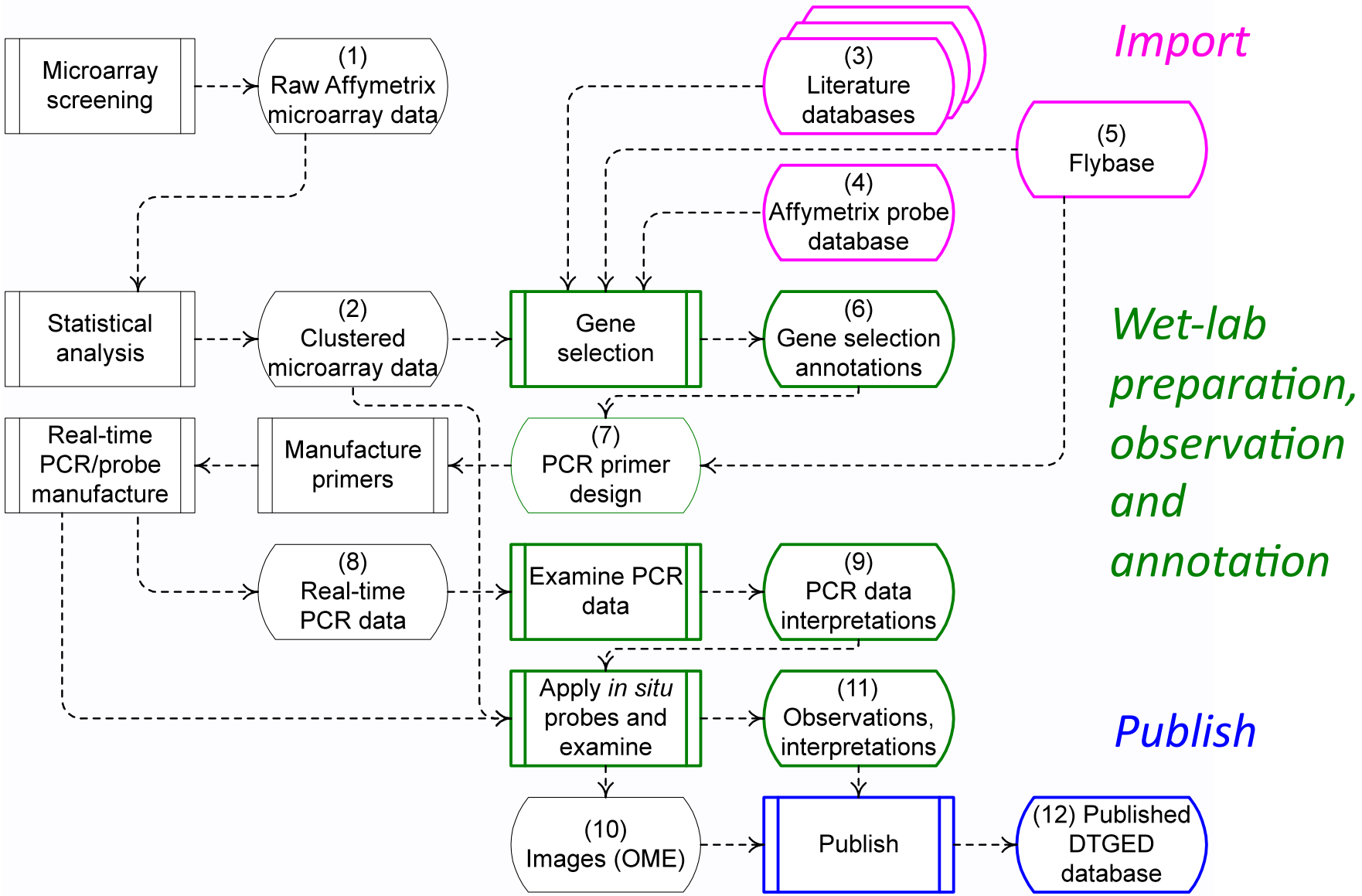


CG12907 aly

CG12907 topi

Images: Dr Helen White-Cooper

Example: Fly-TED data flows



Requirements

R1: Ease of use: quickly create a simple collection
R2: Ease of use: no programming or HTML coding
R3: Ease of use: no knowledge of RDF and/or OWL
R4: Flexibility: choice of RDF vocabulary used
R5: Flexibility: define or adapt structure of data
R6: Sharability: including online access and offline copying
R7: Remixability: linkable, use domain vocabularies
R8: Portability: move data between systems; not centralized
R9: Sustainable software: use unmodified software
R10: Sustainable of data: standard, easily used format
R11: Exposed data: accessible to independent software
R12: Offline working

Survey

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Callimachus	✘	✘	✘	✔	?	?	✔	✘	✧	✔	✧	?
Semantic MediaWiki	✧	✔	✔	✔	✧	✘	✧	?	✔	✘	✧	✘
Wikidata	✘	✔	✔	✔	✧	✘	✔	✘	✔	✘	✧	✘
Protege	✘	✔	✘	✔	✧	✔	✔	✔	✔	✔	✔	✔
Figshare	✔	✔	✔	✘	✘	✘	✘	✘	✔	✘	✔	✘
ResearchSpace	✔	✔	✔	✘	✘	✘	✔	✘	✔	✧	?	✘
Histcross/Segrada	✔	✔	✔	✧	?	✘	✘	✘	✧	✘	✘	✔
Spreadsheet	✔	✔	✔	✔	✔	✧	✘	✔	✔	✧	✔	✔
Rightfield	✔	✔	✧	✔	✧	✧	✧	✔	✔	✧	✔	✔
Desktop database	✘	✧	✘	✔	✧	✘	✘	✔	✔	✘	✘	✔
CMS	✘	✔	✔	✔	✧	✘	✧	✘	✔	✘	✧	✘
ELN	✔	✔	✔	✧	✧	✧	✘	✘	✔	✘	✧	✘
Annalist	✔	✔	✧	✔	✧	✔	✔	✔	✔	✔	✔	✔

✔ Yes ✘ No ✧ Partial ? Unknown

Motivation and Requirements

Annalist in use

Design and implementation

Status and future work

Discussion

View of Photograph

Id	20150501_1644_032	Type	Photograph
Label	20150501-1644-032.jpg Soar/Sileby/Bridge		
Date taken	2015-05-01	Location taken	Sileby mill
Comment	Bridge over weir stream at Sileby Mill on the River Soar 2015/20150501-1644-032.jpg		

Image

Keywords

Keyword
Landscape
River Soar
Sileby Mill
Leicestershire

Location notes

Id	Sileby_mill	Label	Sileby mill	Map reference	SK 592 147

Edit | Copy | Close | Set default view | View description

Choose view: View of Photograph | Show view

JSON-LD DATA

View of Photograph

Home | Photo_collection | Photograph | User gklyne | Logout

Id: 20150501_1644_032

Type: Photograph

Label: 20150501-1644-032.jpg Soar/Sileby/Bridge

Date taken: 2015-05-01

Location taken: Sileby mill

Comment:

Bridge over weir stream at Sileby Mill on the River Soar

2015/20150501-1644-032.jpg

Image: Browse... No file selected.

Previously uploaded: 20150501-1644-032.jpg

Keywords

Keyword
Landscape
River Soar
Sileby Mill
Leicestershire

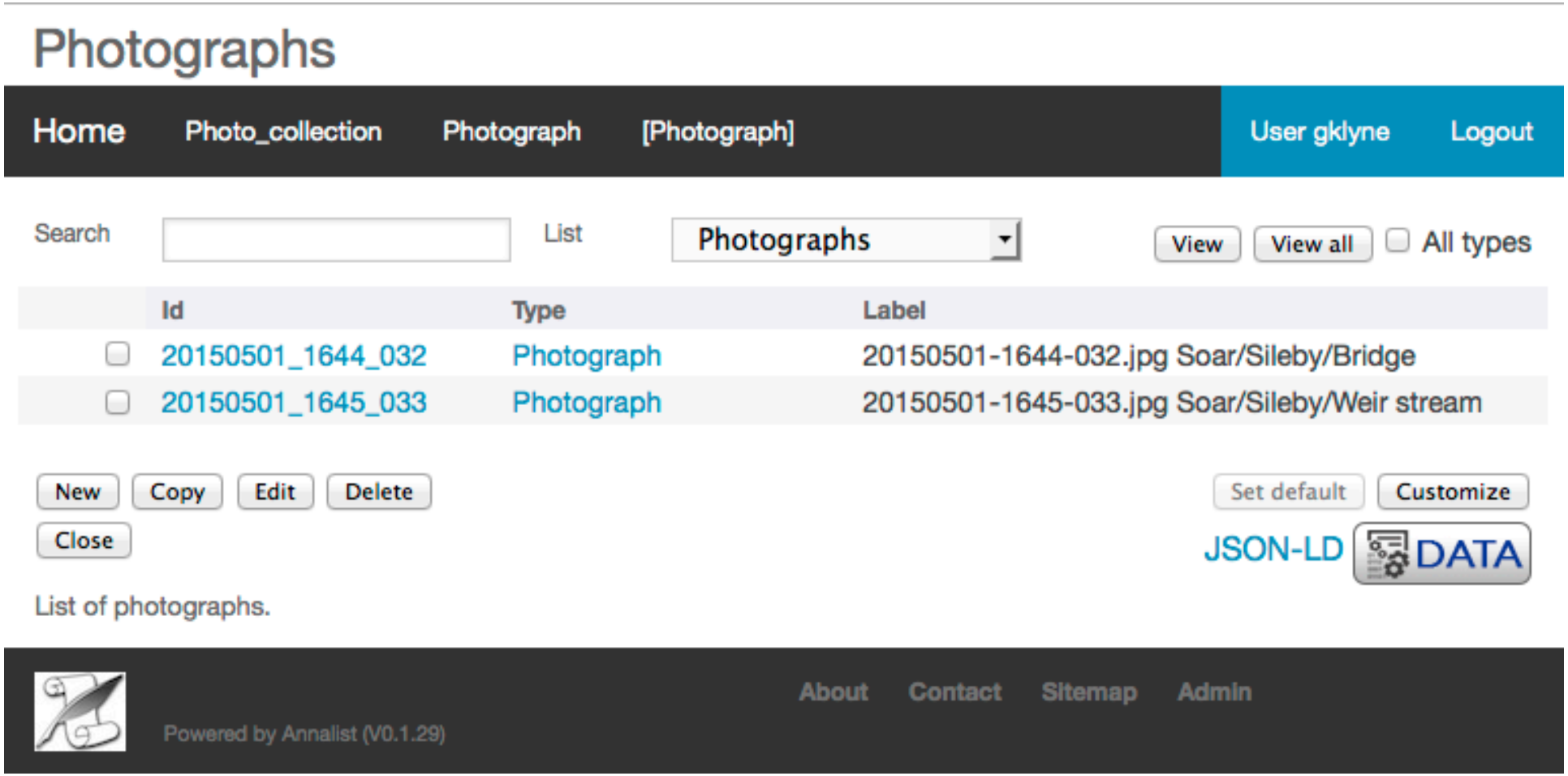
Remove Keyword | Add Keyword | Move ↑ | Move ↓

Location notes: Sileby mill

Save | View | Cancel | Edit view

Choose view: View of Photograph | Show view

View of Photograph



http://annalist.net/

Motivation and Requirements

Annalist in use

Design and implementation

Status and future work

Discussion

Design philosophy

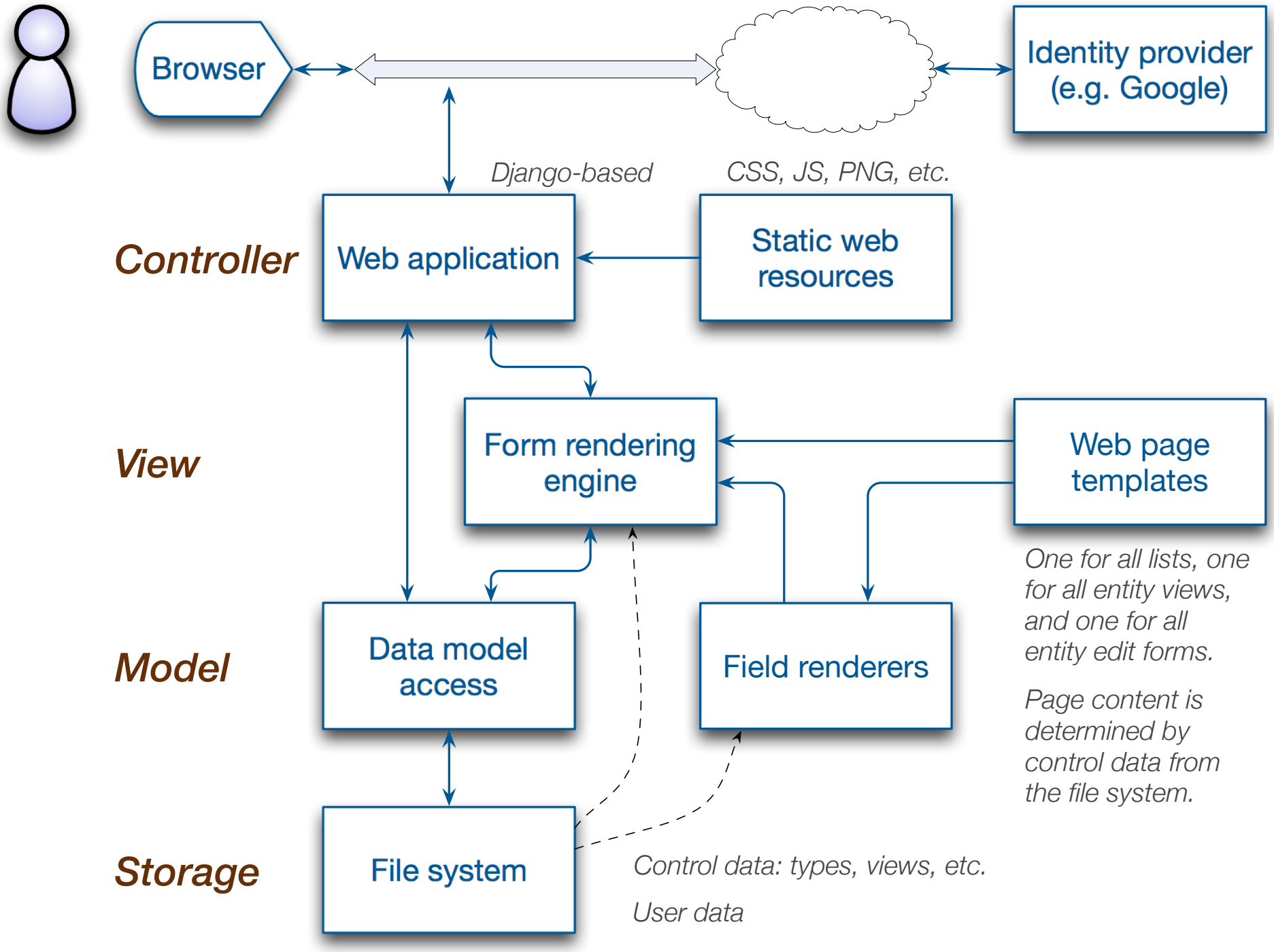
- Data first, structure later
- Minimize impediments to data entry
- Annalist as a part in a wider linked data ecosystem
- JSON-LD as “view source” for linked data

The message that data can tell is not always clear at the outset, but may emerge though the process of observation and collection.

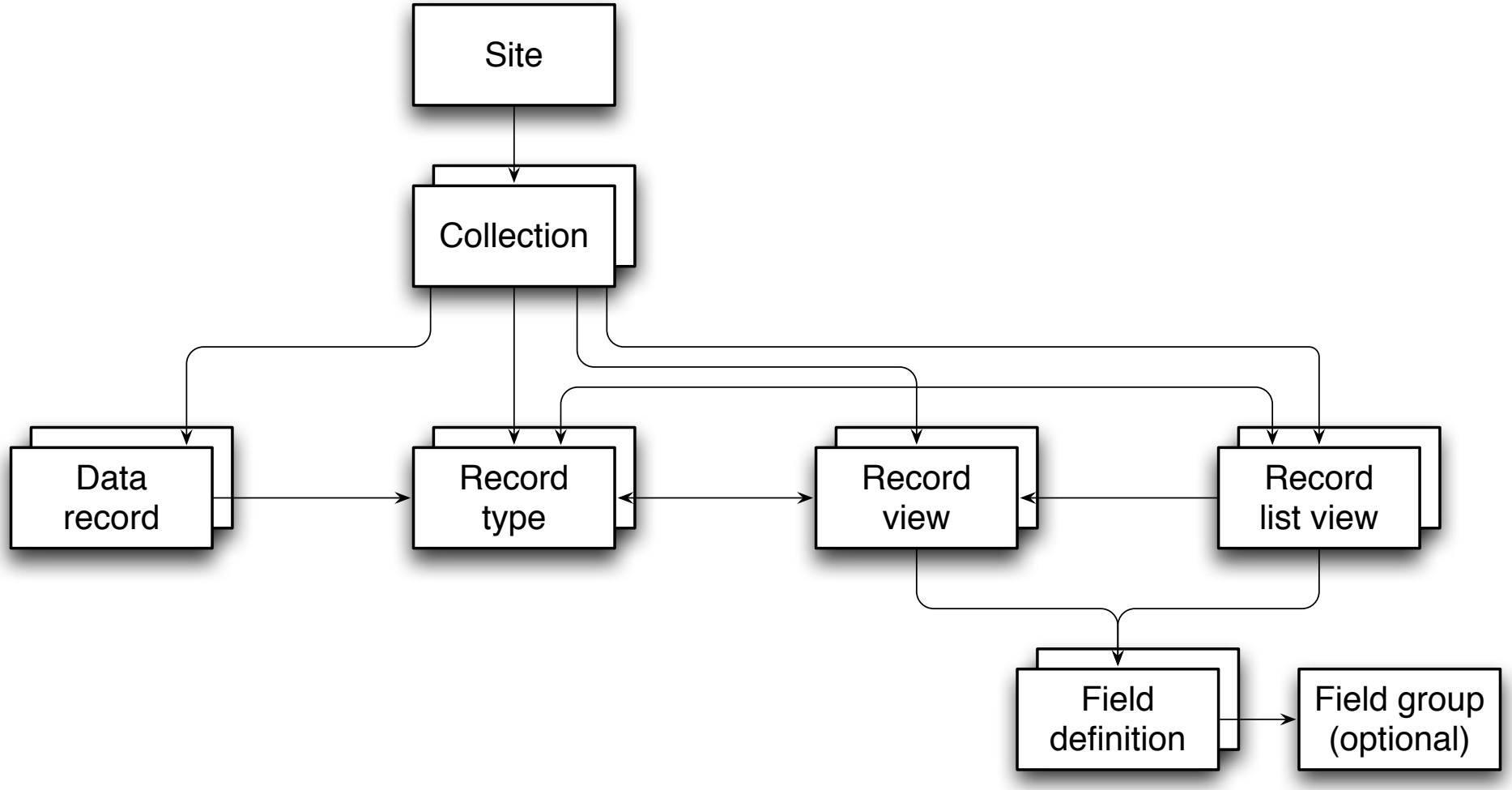
Technical implementation

- Web server application
- Deployable in desktop, workgroup or cloud
  - (Docker too)
- Main logic is server-based
  - Javascript for responsive interface

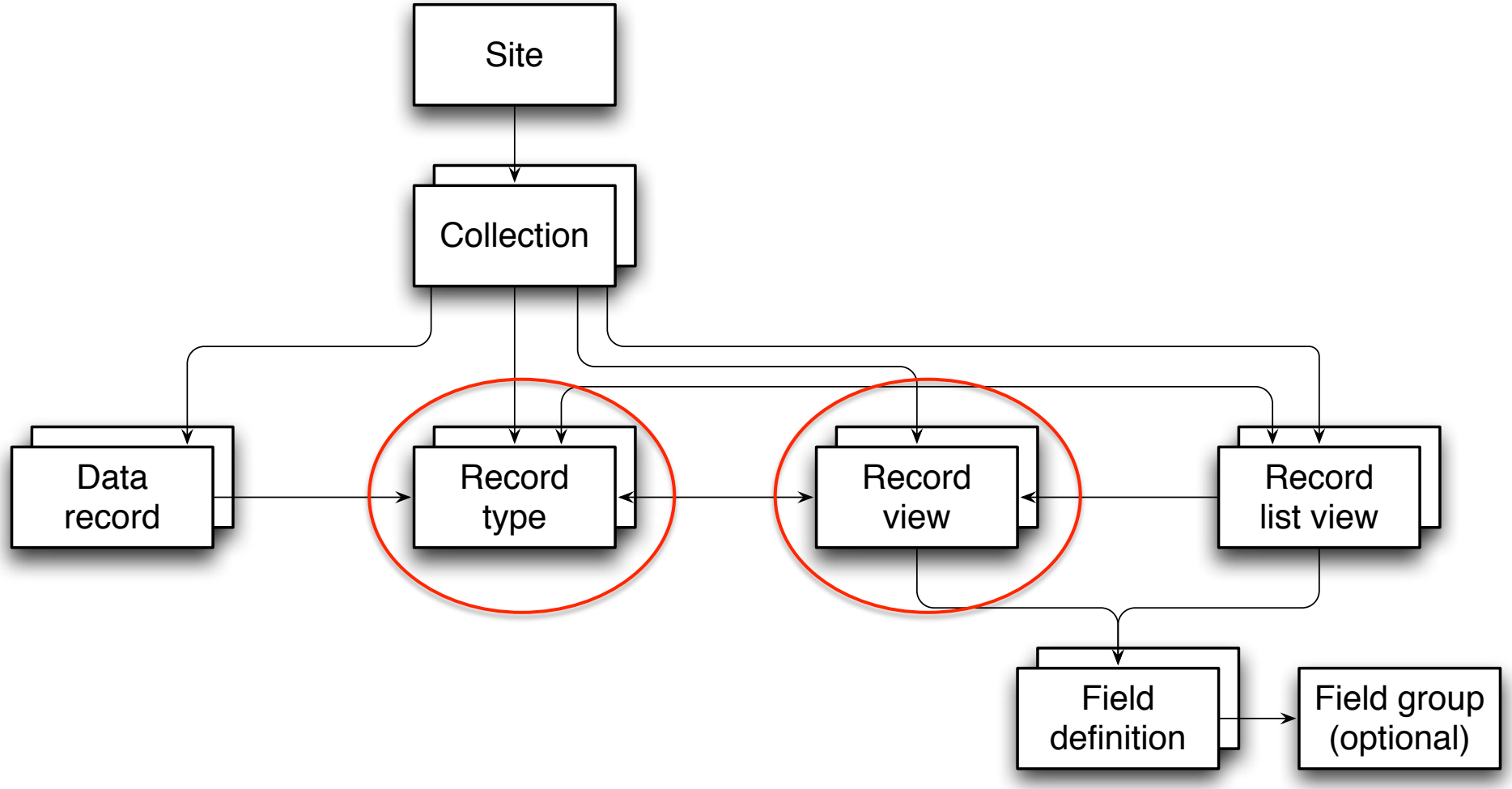
- Data stored as JSON-LD files
  - Can be published directly by HTTP server
- Customizable form generator
  - Definition as JSON-LD, also managed by Annalist
  - One definition used for view and edit forms


Browser
Identity provider (e.g. Google)
Django-based
CSS, JS, PNG, etc.
Controller
Web application
Static web resources
View
Form rendering engine
Web page templates
One for all lists, one for all entity views, and one for all entity edit forms.
Model
Data model access
Field renderers
Page content is determined by control data from the file system.
Storage
File system
Control data: types, views, etc.
User data

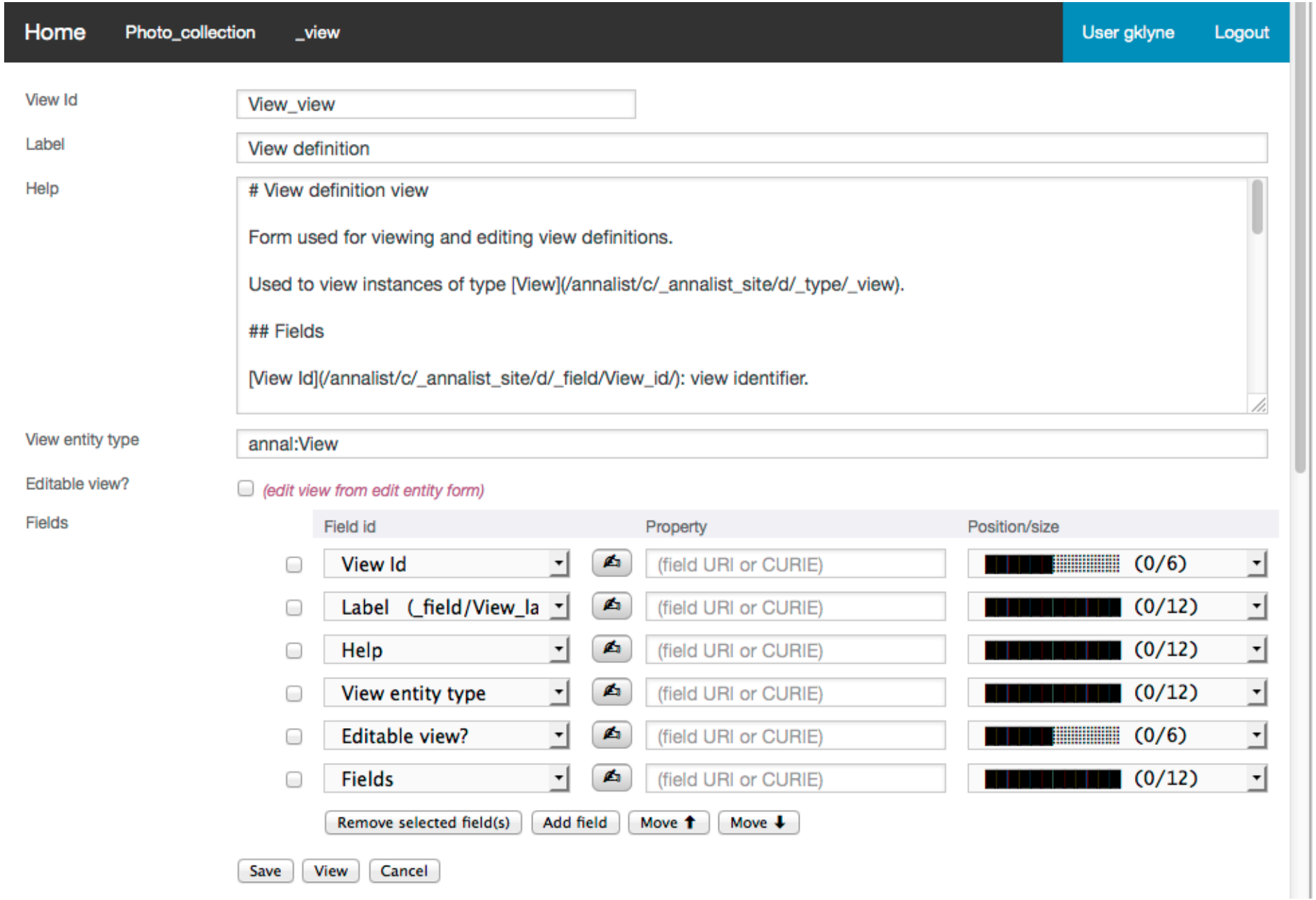
Data model concepts



Data model concepts



Configuration self-maintenance



JSON-LD (“view source”)

```
{ "@id":                "annal:display/View_view"
, "@type":              ["annal:View"]
, "@context":           ["../../coll_context.jsonld"]
, "annal:id":           "View_view"
, "annal:type_id":      "_view"
, "annal:uri":          "annal:display/View_view"
, "annal:record_type":  "annal:View"
, "rdfs:label":         "View definition"
, "rdfs:comment":       "# View definition view\r\n\r\nForm used for viewing ...
, "annal:open_view":    false
, "annal:view_fields":
  [ { "annal:field_id":              "_field/View_id"
    , "annal:field_placement":       "small:0,12;medium:0,6” }
  , { "annal:field_id":              "_field/View_label"
    , "annal:field_placement":       "small:0,12” }
  , { "annal:field_id":              "_field/View_comment"
    , "annal:field_placement":       "small:0,12” }
  , { "annal:field_id":              "_field/View_target_type"
    , "annal:field_placement":       "small:0,12” }
  , { "annal:field_id":              "_field/View_edit_view"
    , "annal:field_placement":       "small:0,12;medium:0,6” }
  , { "annal:field_id":              "_field/View_fields"
    , "annal:field_placement":       "small:0,12” }
  ]}
```

Motivation and Requirements

Annalist in use

Design and implementation

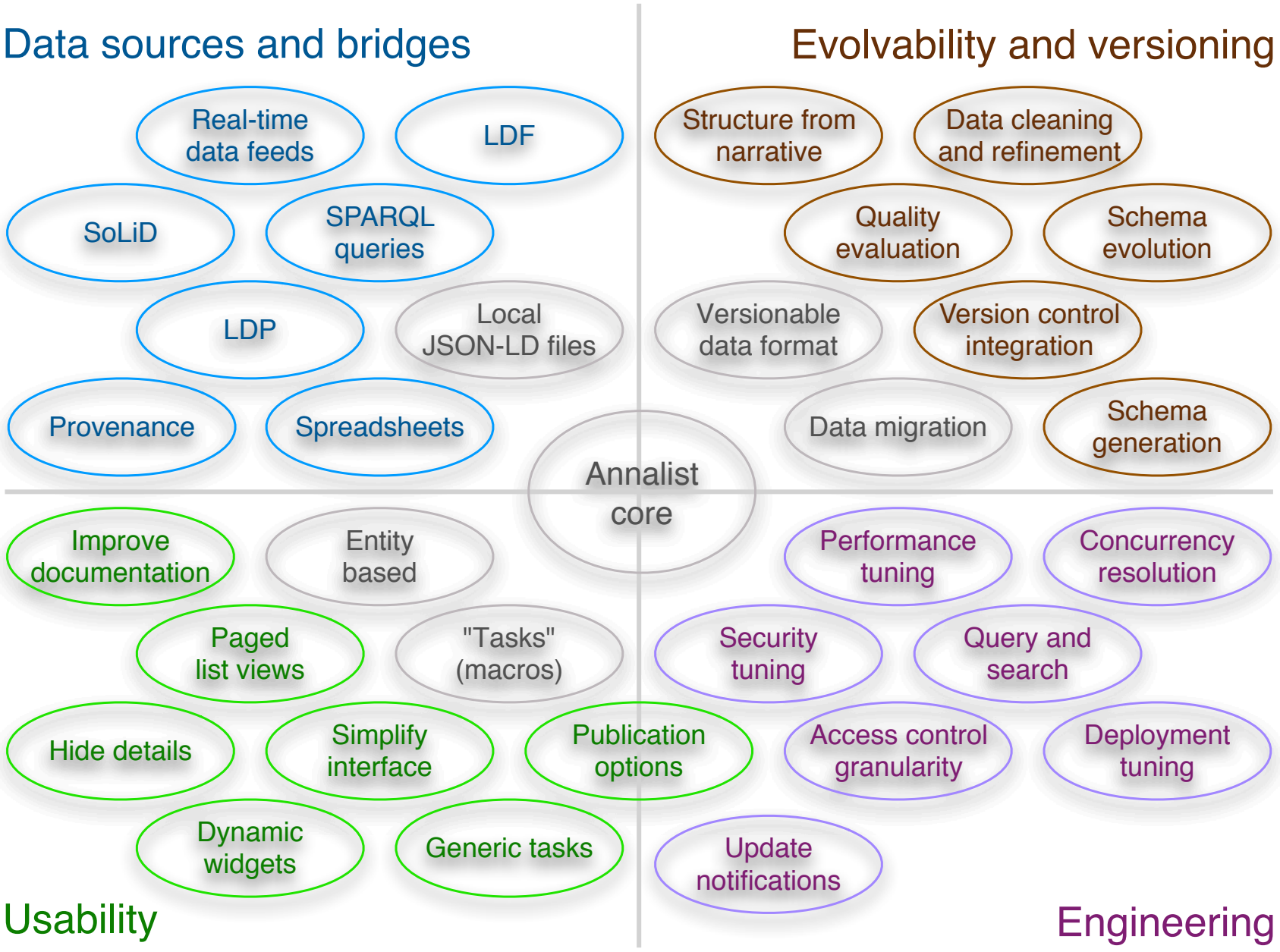
Status and future work

Discussion

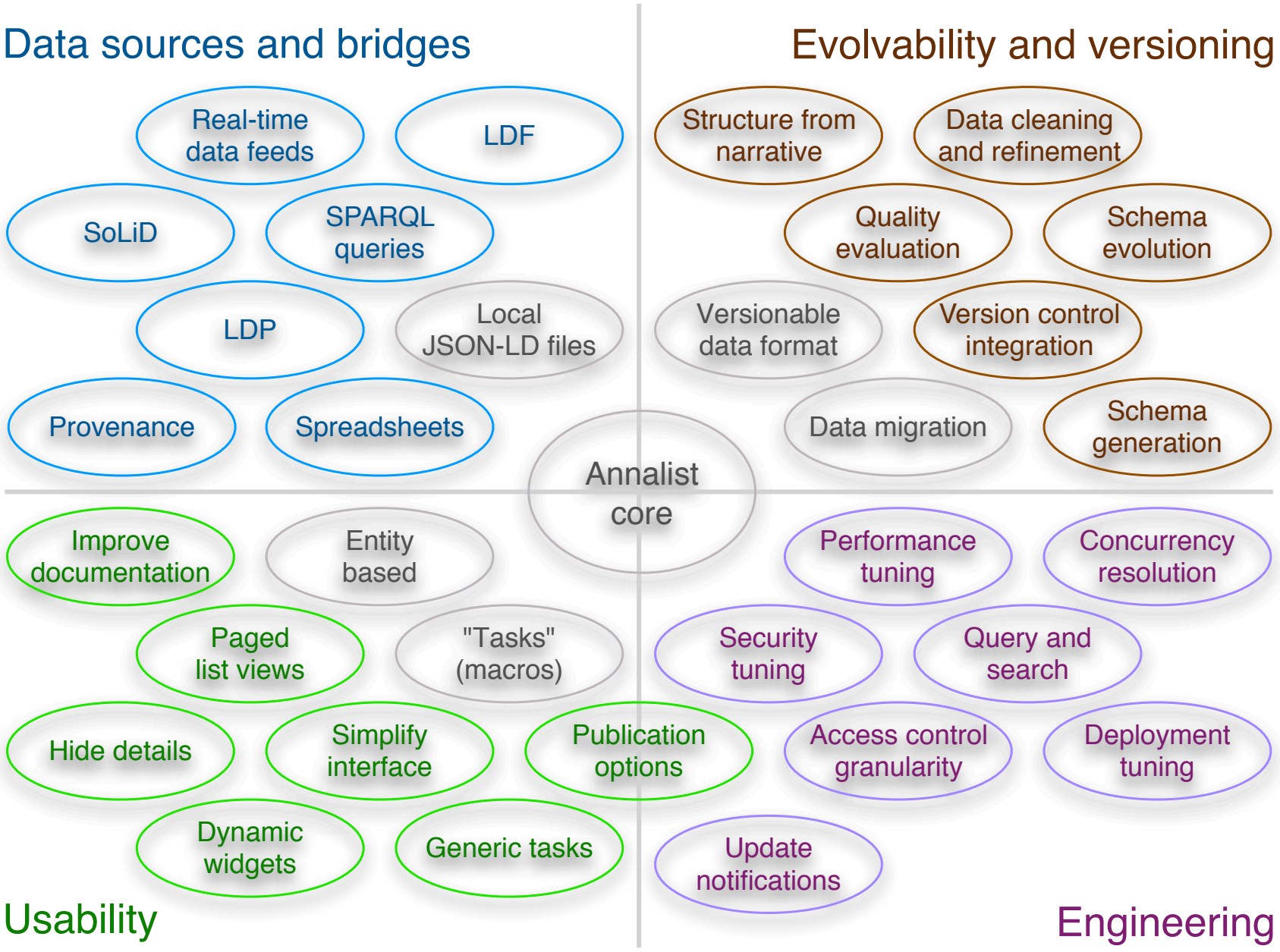
Progress to date

1. A viable tool to create and share linked data
2. Flexible to deal with diverse applications
3. Robust
    - even as a work-in-progress, I have never lost application data due to an Annalist software fault
4. At least approachable for users who are not familiar with RDF

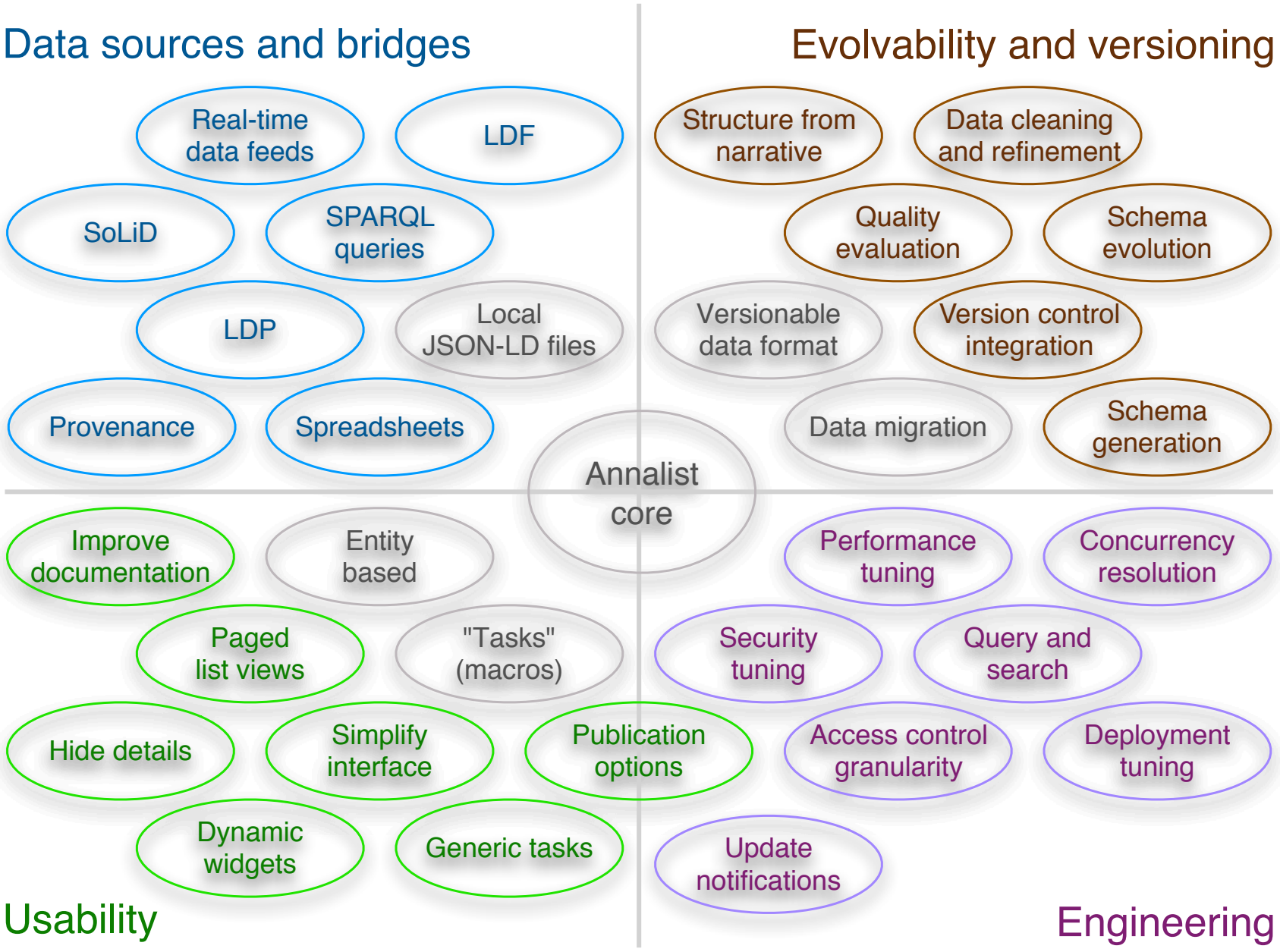
Status and future work



Status and future work



Status and future work



Transition to community project

A public repository is a start:

- https://github.com/gklyne/annalist
- MIT licence

But there remain many things to do...

- Governance
- Supporting documentation
- Engage other developers
- Integration with complementary systems
- Application data definition “libraries”

Discussion

Some topics

Ontologies optional (schema follows data)?

Locators and Identifiers (local vs global names)?

Evolution and versioning: in/of data? Git or PROV?

Primacy of source (JSON) over data model (triples)?

Some issues raised by paper reviewers:

- Evolution: data or schema
- Concurrent access conflicts
- Access control granularity
- Usability, evaluation
- Flat file storage performance

Links

http://annalist.net

Paper

https://github.com/gklyne/annalist/blob/develop/documents/publications/LDOW2016-paper/Annalist-paper-ACMSIG.pdf

Slides

https://github.com/gklyne/annalist/blob/develop/documents/publications/LDOW2016-paper/annalist-presentation-ldow2016.pptx, .pdf

Demo site

http://demo.annalist.net/

Tutorial

http://annalist.net/documents/tutorial/annalist-tutorial.html

Software

https://github.com/gklyne/annalist/

Evolvability

Recall: data first philosophy

Two aspects of evolution:

- adding structure to data (add schema)
- changing existing structure (schema change)

Types vs properties:

- using Annalist, types primarily affect resource naming (entity names)
- properties affect content (JSON keys)

Data migration

Focus on type and property URI changes

Adopting a guided approach for now

```
$ annalist-manager migratecollection Performance_defs Journal_defs

# Migration report from collection 'Performance_defs' to 'Journal_defs' #

  * Type Uploaded_audio, URI changed from 'coll:Uploaded_audio' to
    'coll:Uploaded_audio_test'
      Consider adding supertype 'coll:Uploaded_audio' to
          type 'Uploaded_audio' in collection 'Journal_defs'
       URI 'coll:Uploaded_audio' appears as entity type for view 'Uploaded_audio'
       URI 'coll:Uploaded_audio' appears as entity type for list 'Uploaded_audio'
       URI 'coll:Uploaded_audio' appears in selector for list 'Uploaded_audio'
       URI 'coll:Uploaded_audio' appears as entity type for group Uploaded_audio_m
       URI 'coll:Uploaded_audio' appears as entity type for group Uploaded_audio_r
  * Field Linked_audio, property URI changed from 'coll:audio_clip' to 'coll:linked_audio'
      Consider adding property alias for 'coll:audio_clip' to type
          Linked_audio in collection 'Journal_defs'
  * Field Web_resource, property URI changed from 'coll:web_resource' to 'coll:resource'
      Consider adding property alias for 'coll:web_resource' to type
          Web_resource in collection 'Journal_defs'
```

Concurrent access conflicts

Atomic updates to single entity

Design to detect update conflicts:

- detect changes while an edit is in progress
- cf. HTTP entity tag (ETag)
- not currently implemented

No consistency checks between entities

- storage model doesn't care about consistency
- consider as aspect of data quality checks
- handle post-acquisition, as needed

Access control granularity

Currently:

- control applied per-collection
- permissions associated with authenticated user Id in Annalist “user permissions” record
- limited possibility to require different permissions for different record types

Possibilities:

- type-based permission requirements could offer finer granularity (but not to individual statement level)
- Generalize Annalist trust/permission model for RBAC

Would like:

- To devise way to use OpenID Connect (OAuth2) authentication with WebID permissions; e.g. to work with SoLiD servers

Usability, evaluation

No formal usability study (yet)

- What to test?
- Different user applications are ... different
- How to formally test flexibility?

Evolving interface through experience

- Incremental development, informed by “agile”
- Using Annalist in diverse applications
- Modifying user interface in response to problems experienced

Motivation and Requirements
Annalist in use
Design and implementation
Status and future work
Discussion